МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«Харківський політехнічний інститут»

ІНФОРМАТИКА.
ЧАСТИНА 1

Лабораторний практикум

для студентів
6.050502 «Інженерна механіка»,
6.050403 «Інженерне матеріалознавство»

Харків 2016

УДК 004.43
ББК 32.973-018
Ч-49

Рецензенти:

І. А. Фурман, д-р техн. наук, проф. ХНТУСГ (м. Харків)

М. Й. Заполовський, канд. техн. наук, проф. НТУ «ХПІ» (м. Харків)

Публікується за рішенням вченої ради університету,
Протокол №13 від 25. 06. 2015 р.

Ч34 **Інформатика**: лабораторний практикум. ч. 1. /О.П. Черних, О.М. Шеін. – Х.: НТУ «ХПІ», 2016. – 110 с.

Наведена теоретична частина для розв'язання задач з використанням мови програмування високого рівня Pascal. Описано вісім лабораторних робіт з формулюванням завдання, схемами алгоритмів, текстами програм з докладними коментарями та отриманими результатами.

Призначено до виконання лабораторних робіт з навчальної дисципліни «Інформатика» для студентів спеціальності 6.050502 «Інженерна механіка», 6.050403 «Інженерне матеріалознавство».

Іл. 25. Табл. 26. Бібліогр. 6 назв.

УДК 004.43
ББК 32.973-018

ВСТУП

У наш час триває процес все ширшого впровадження комп'ютерних технологій у всі сфери діяльності та повсякденного життя людини. До недавнього минулого дещо в стороні від цього процесу залишалася така важлива професійна галузь, як машинобудування. В останній час ситуація суттєво змінилася. Впровадження засобів обчислювальної техніки кардинально змінило вже існуючі напрями, а також сприяло виникненню нових методик проектування, виробництва та експлуатації виробів машинобудівної галузі.

На механіко-технологічному факультеті НТУ «ХПІ» в рамках підготовки бакалаврів викладається навчальна дисципліна «Інформатика», програма якої розроблена співавтором цього видання доцентом, кандидатом фізико-математичних наук Черних О.П.

Перша частина даного лабораторного практикуму є додатком до теоретичного курсу і призначена для надбання студентами практичних знань та досвіду у розв'язанні широкого кола задач з використанням мови програмування високого рівня Pascal. При цьому студенти виконують всі етапи, починаючи з розробки алгоритму розв'язання задачі та закінчуючи написанням й запуском програми, яка цей алгоритм реалізує.

Перша частина практикуму включає в себе вісім розділів.

У першому розділі «РОЗВ'ЯЗАННЯ НАЙПРОСТІШИХ ЗАДАЧ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ Pascal» розглянуті запитання алгоритмізації задач. Наведено структуру програми мовою програмування Pascal, класифікацію основних типів даних та операцій над ними, структуру основних операторів програми, а також вичерпний перелік стандартних процедур та функцій для роботи з різними типами даних.

Другий розділ «ПРОГРАМУВАННЯ ВКЛАДЕНИХ ЦИКЛІВ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ Pascal» присвячений

розв’язанню циклічних задач з використанням різних видів операторів циклу.

У третьому розділі «РОБОТА З ФАЙЛАМИ ТА МАСИВАМИ» наведено класифікацію файлів та набір процедур і функцій для роботи з ними. Також розглянуто роботу із структурованими даними типу масив.

Четвертий розділ «РОБОТА З МАТРИЦЯМИ» розглядає використання багатовимірних масивів.

П’ятий розділ «ПРОЦЕДУРИ ТА ФУНКЦІЇ» присвячений запитанням структурного програмування та розглядає можливості користувача у створенні власних процедур та функцій.

У шостому розділі «ВЛАСНІ МОДУЛІ В TP» розглядаються можливості користувача зі створення бібліотеки, що складається із власних процедур та функцій.

Сьомий розділ «ЗАПИСИ В TP» розглядає використання структурованих типів даних, що містять у собі компоненти різних типів.

Восьмий розділ «СТАНДАРТНІ МОДУЛІ В TP» присвячений використанню можливостей стандартних бібліотек для роботи у текстовому та графічному режимах.

Всі розділи виконані в єдиному стилі. Кожен з них включає в себе такі пункти:

- назва роботи;
- мета виконання роботи;
- загальні положення, де у повному обсязі викладені теоретичні відомості, необхідні для виконання роботи;
- варіанти індивідуальних завдань;
- приклад виконання лабораторної роботи, який включає: формулювання завдання, схему алгоритму, текст програми з докладними коментарями, отримані результати.

Обсяг інформації, наведеної у загальних положеннях та прикладах виконання робіт, є достатнім для виконання індивідуальних завдань.

ЛАБОРАТОРНА РОБОТА 1

РОЗВ'ЯЗАННЯ НАЙПРОСТІШИХ ЗАДАЧ ІЗ ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ Pascal

Мета роботи − набуття практичних навичок у розробці та виконанні найпростіших задач мовою програмування Pascal.

Загальні положення

1.1. Алгоритмізація завдань

1.1.1. Порядок викоання завдань на ПК

Процес створення програми для розв'язання будь-якої практичної задачі складається з таких етапів:

1) Постановка завдання й визначення кінцевих цілей.

Завдання формулюється користувачем або отримується ним у вигляді завдання.

2) Математичне формулювання задачі.

На цьому етапі здійснюється формалізація завдання. При цьому необхідно обрати один з відомих методів математичного опису задач або розробити новий спосіб, якщо жоден із існуючих методів не може бути використаний. Визначаються вхідні дані, початкові умови, точність обчислень. При побудові математичних моделей необхідно її грамотне спрощення шляхом ігнорування несуттєвих факторів.

3) Вибір числового методу.

Для більшості завдань, що зустрічаються на практиці, точні методи або невідомі, або громіздкі. Для вирішення цих задач використовуються числові методи, які забезпечують виконання завдання із заданою точністю. Як правило, для розв'язання однієї і тієї ж задачі можуть бути використані різні числові методи. Вибір числового методу залежить від багатьох чинників: простоти програмної

реалізації, точності, часу виконання завдання. Найчастіше такі вимоги є суперечливими. Наприклад, збільшення кількості ітерацій підвищує точність, але збільшує час виконання завдання.

4) Розробка алгоритму розв'язання задачі.

Алгоритм складається відповідно до обраного методу розв'язання задачі. Для розв'язання однієї і тієї ж задачі може бути складено кілька різних алгоритмів. При цьому обирається оптимальний, який забезпечує найбільш ефективне використання ресурсів ПК. Хоча алгоритм не орієнтується на конкретний тип ЕОМ і характеризує метод розв'язання задачі.

5) Конструювання програми.

Запис програми алгоритмічною мовою. При цьому залежно від типу задачі, що розв'язується, можуть бути використані різні стилі, технології та методології програмування.

Вимогами до програми є: простота для розуміння і відлаження; ефективне використання комп'ютерних ресурсів і можливість швидкого виконання.

6) Введення тексту програми і вхідних даних.

Текст програми набирається з використанням текстового редактора. Іноді вхідні дані зручно помістити в окремий файл.

7) Компіляція, наладження і виконання програми.

При складанні програми необхідно враховувати логічні обмеження, які можуть призвести до переповнення розрядної сітки або неможливості подальшого виконання програми (ділення на нуль, корінь парного степеня і логарифм від'ємного числа та ін.). Це повинно бути передбачено у програмі у вигляді перевірки деяких умов.

8) Отримання рішення і аналіз результату.

Результат може бути отриманий у вигляді чисел, таблиць, графіків та ін. Результат може бути виведений на екран або у файл.

9) Обробка та аналіз результату.

1.1.2. Алгоритми

Алгоритм − кінцеве число директив, що однозначно визначають процес перетворення вхідних даних у кінцевий результат.

Властивості алгоритму:

− масовість − придатність для виконання завдань даного класу, незалежно від вхідних даних;

− визначеність − всі інструкції повинні виконуватися чітко і не допускати двозначних трактувань;

− однозначність (детермінованість) результатів − результат повинен збігатися при багаторазовому виконанні завдання;

− результативність − виконання алгоритму повинно приводити до отримання результату;

− кінцевість − процес виконання завдання повинен закінчуватися за кінцевою кількістю кроків.

Існує кілька способів опису алгоритмів:

− словесний − подає алгоритм у вигляді слів і пропозицій;

− табличний − використовується для опису складних логічних умов, що визначають ті чи інші обчислення;

− операторний − алгоритм подається у вигляді послідовності операторів;

− схемний (графічний) − алгоритм задається у вигляді схеми.

В інженерній практиці найбільшого поширення набув схемний спосіб опису алгоритмів, при цьому процес обчислень розподіляється на окремі операції, які відображаються у вигляді умовних графічних блочних символів. Блоки зазвичай мають наскрізну нумерацію.

Правила виконання схем, перелік блоків, їх найменування, форма і розміри встановлюються ГОСТом, який відповідає міжнародному стандарту UCO. Програмні документи оформляються відповідно до стандартів:

– схеми алгоритмів для документів ГОСТ 19.701-90;

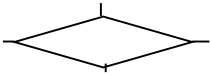
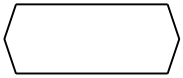
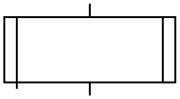
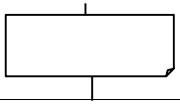
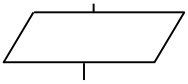
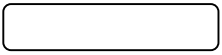
– ЕСКД − Єдина система конструкторської документації;

– ЕСПД − Єдина система програмної документації;

– діаграмами міжнародного стандарту UML.

Блоки, що найбільш часто вікористовуються, наведені у табл. 1.1.

Таблиця 1.1 – Блоки алгоритмів

Назва символу	Символ	Примітка
Процес		Обчислювальна дія або послідовність дій
Рішення		Перевірка умов
Модифікація		Заголовок циклу
Зумовлений процес		Обчислення за підпрограмою
Документ		Виведення даних на друк
Введення/ Виведення		Введенння/Виведення даних (незалежно від фізичного носія інформації)
Пуск/Зупинка		Початок, кінець алгоритму, вхід і вихід у підпрограмах
Міжрядковий з’єднувач		Розрив ліній потоку в межах однієї сторінки
Міжсторінковий з’єднувач		Розрив ліній потоку, розташованих на різних сторінках
Коментарі		Пояснення блоків і підпрограм
Лінії		Указують на хід процесу

1.1.3. Типові структури алгоритмів

За характером зв'язків між блоками розрізняють такі типові структури алгоритмів: лінійна, розгалужена і циклічна структури.

– алгоритм лінійної структури (лінійний алгоритм) − алгоритм, в якому блоки виконуються послідовно один за одним. Такий порядок виконання блоків називається природним;

– алгоритм розгалуженої структури (розгалужений алгоритм) − алгоритм, в якому обчислювальний процес йде по одній або іншій гілці залежно від виконання певних умов;

– алгоритм циклічної структури (циклічний алгоритм) − алгоритм, в якому визначена послідовність дій повторюється багато разів при різних значеннях величин, що входять до них. Багаторазово повторювані ділянки називаються циклами;

– алгоритм із структурою вкладених циклів (циклічний алгоритм) − це алгоритм, в якому всередині одного циклу, що називається зовнішнім (глобальним), розташований інший цикл, який називається внутрішнім (локальним).

1.2. Структура програми

Програма мовою Pascal складається з заголовка і блока (тіла програми), за яким прямує точка, що є ознакою кінця програми.

Загальна структура програми має вигляд:

```
PROGRAM <имя>; {Заголовок програми}
  LABEL <метка>,…,<метка>; {Розділ опису міток}
  CONST  {Розділ опису констант}
     <имя конст.>=<конст.>;
            . . .
     <имя конст.>=<конст.>;
  TYPE   {Розділ опису типів }
     <имя типа>=<тип>;
            . . .
     <имя типа>=<тип>;
```

```
VAR  {Розділ опису змінних}
     <имя перем.>,…<имя перем.>:<тип>;
                  . . .
     <имя перем.>,…<имя перем.>:<тип>;
PROCEDURE <имя>(<сп. форм. пар.>); {Розділ опису процедур}
          <блок>;
FUNCTION <имя>(<сп. форм. пар.>):<тип>; {Розділ опису
функцій}
          <блок>;
BEGIN  Розділ операторів}
    <оператор>;
       . . .
    <оператор>;
END.
```

Заголовок програми містить ключове слово PROGRAM, за яким прямує ім'я програми. Ім'я програми має відображати головну функцію програми і не може бути використано всередині програми. У TP заголовок ігнорується компілятором і є декоративною частиною програми. Програму можна писати рядковими чи великими буквами або використовуючи їх комбінацію.

Блок (тіло) програми містить розділи описів (декларативна частина) і розділ операторів (частина, що виконується).

Розділ описів може включати:

- розділ опису міток;
- розділ опису констант;
- розділ опису типів;
- розділ опису змінних;
- розділ опису процедур;
- розділ опису функцій.

Стандарт мови Паскаль регламентує наведений вище порядок проходження розділів описів. Виходячи з міркувань хорошого стилю програмування, доцільно дотримуватися вимог стандарту.

1.3. Класифікація типів даних

1.3.1. Цілі типи

Є п’ять цілочислових типів, діапазон можливих значень яких залежить від їх внутрішнього подання. Типи цілих даних, обсяг займаної пам’яті і діапазон можливих значень наведено у табл. 1.2.

Таблиця 1.2 – Цілі типи

Тип	Довжина, байт	Діапазон
INTEGER	2	− 32 768..32767
SHORTINT	1	−128 .. 127
LONGINT	4	−2 147 483 648.. 2 147 483 647
BYTE	1	0..2555
WORD	2	0..65535

Над даними цілого типу визначені такі операції:

+, -, /, *, div, mod, =, <>, <,>, <=,> =

Набір убудованих математичних функцій, які можна застосувати до даних цілого типу, наведено у табл. 1.3.

Таблиця 1.3 − Математичні функції для даних цілого типу

Звернення	Призначення
abs(x)	Повертає модуль x
sqr(x)	Повертає квадрат x
random(w)	Генерує псевдовипадкове число на інтервалі. Повертає модуль 0 <x ≤w
odd(i)	Повертає true, якщо і непарне
chr(b)	Повертає символ за кодом і
dec(x, k)	Зменшує x на k або 1
inc(x, k)	Збільшує x на k або 1
hi(i)	Повертає старший байт
lo(i)	Повертає молодший байт
swap(i)	Змінює байти місцями

Примітка: x − будь-який цілий тип; i – integer тип; b – byte тип; w − word тип.

1.3.2 Дійсні типи

Є п’ять видів дійсних типів, діапазон можливих значень яких залежить від їх внутрішнього подання. Типи, обсяг займаної пам’яті, кількість значущих цифр і діапазон можливих значень наведено у табл. 1.4.

Таблиця 1.4 − Дійсни типи

Тип	Довжина, байт	Кількість знач. цифр	Діапазон
REAL	6	11..12	$2{,}9\cdot10^{-39}..1{,}7\cdot10^{-38}$
SINGLE	4	7..8	$1{,}5\cdot10^{-45}..3{,}4\cdot10^{-38}$
DOUBLE	8	15..16	$5{,}0\cdot10^{-324}..1{,}7\cdot10^{-308}$
EXTENDED	10	19..20	$3{,}4\cdot10^{-4951}..1{,}1\cdot10^{-4932}$
COMP	8	19..20	$2^{63}+1..2^{63}-1$

Над даними дійсного типу визначені такі операції:

+, -, /, *, =, <>, <,>, <=,> =

Набір убудованих математичних функцій, які можна застосувати до даних дійсного типу, наведено у табл. 1.5.

Таблиця 1.5 − Математичні функції для даних дійсного типу

Звернення	Призначення	Приклад
1	2	3
sin(x)	Повертає синус, кут в радіанах	
cos(x)	Повертає косинус, кут в радіанах	
arctan(x)	Повертає арктангенс	
abs(x)	Повертає абсолютну величину	
sqr(x)	Повертає квадрат	
exp(x)	Повертає експоненту	
ln(x)	Повертає натуральний логарифм	
trunc(x)	Відкидає дробову частину	trunc(5.6) = 5

Продовження табл.1.5

1	2	3
round(x)	Округлює до найближчого цілого	round(5.6) = 6
frac(x)	Виділяє дробову частину	frac(10.1) = 0.1
int(x)	Виділяє цілу частину	int(10.1) = 10.0
random	Датчик випадкових чисел (ДВЧ) 0 <n ≤1	
random(x)	Датчик випадкових чисел (ДВЧ) 0 <n ≤ x	
randomize	Ініціалізація ДВЧ	

1.3.3. Логічний тип

Дані логічного типу можуть набувати значення однієї з двох логічних констант: FALSE; TRUE.

Діапазон значень: FALSE. . TRUE.

Логічні значення мають порядковий номер 0 і 1.

До логічних (булевих) операндів застосовуються такі операції:

not, and, or, xor

і співвідношення:

=, <>, <,>, <=,> =

Результати операцій над даними логічного типу подаються у вигляді таблиці істинності (табл. 1.6).

Таблиця 1.6 − Таблиця істинності для даних логічного типу

A	B	not A	A and B	A or B	A xor B
1	−	1	−	−	−
0	−	0	−	−	−
1	1	−	1	1	0
1	0	−	0	1	1
0	1	−	0	1	1
0	0	−	0	0	0

Набір убудованих математичних функцій, які можна застосувати до даних логічного типу, наведено у табл. 1.7.

Таблиця 1.7 − Математичні функції, які застосовуються для даних логічного типу

Звернення	Призначення	Приклад
ord(x)	Повертає порядковий номер	ord(False) =True
succ(x)	Повертає наступне значення	succ(False) = True
pred(x)	Повертає попередне значення	pred(True) = False
odd(x)	Перевірка на непарність. True - якщо x непарне	
eoln(x)	True – якщо досягнуто кінець рядка	
eof(x)	True – якщо досягнуто кінець файлу	

1.3.4. Символьний тип

Символьний тип є впорядкованою множиною розширеного набору символів коду ASCII.

Кожному символу коду ASCII відповідає число від 0 до 255, яке є кодом його внутрішнього подання. У табл. 1.8 наведено символи коду ASCII.

Таблиця 1.8 - Символи коду ASCII

Код	Символ	Код	Символ	Код	Символ	Код	Символ
1	2	3	4	5	6	7	8
0	NUL	32	BL	64	@	96	‘
1	SOH	33	!	65	A	97	a
2	STX	34	“	66	B	98	B
3	ETX	35	#	67	C	99	C
4	EOT	36	$	68	D	100	D
5	ENQ	37	%	69	E	101	E
6	ACK	38	&	70	F	102	F
7	BEL	39	‘	71	G	103	G
8	BS	40	(	72	H	104	H

Продовження табл. 1.8

1	2	3	4	5	6	7	8
9	HT	41	)	73	I	105	I
10	LF	42	*	74	J	106	J
11	VT	43	+	75	K	107	L
12	FF	44	,	76	L	108	L
13	CR	45	−	77	M	109	M
14	SO	46	.	78	N	110	N
15	S1	47	/	79	O	111	O
16	DEL	48	0	80	P	112	P
17	DC1	49	1	81	Q	113	Q
18	DC2	50	2	82	R	114	R
19	DC3	51	3	83	S	115	S
20	DC4	52	4	84	T	116	T
21	NAK	53	5	85	U	117	U
22	SYN	54	6	86	V	118	V
23	ETB	55	7	87	W	119	W
24	CAN	56	8	88	X	120	X
25	EM	57	9	89	Y	121	Y
26	EOF	58	:	90	Z	122	Z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	Δ

Код ASCII є семибітним і дозволяє закодувати 27 = 128 символів і становить основний набір коду. У ТР використовується восьмибітний код, який подає закодувати 2^8 = 256 символів, який представляє розширений набір символів.

Символи з кодами 128 .. 255 складають додатковий набір і містять символи національних клавіатур та інші символи додаткового набору.

Символи можна записувати, вказуючи їх код # C, де C належить 0 .. 255.

Наприклад:

65 => 'A'; # 66 => 'B'; # 48 => '0'; # 49 => '1'; # 49 => '#'; # 46 => '.'.

Символи з кодами 0 .. 31 є керівними, вони не відображаються на екрані, а виконують певні функції (деякі з них наведені у табл. 1.9).

Таблиця 1.9 − Керівні символи

Код	Символ	Призначення
#7	BEL	Звуковий сигнал
#9	HT	Горизонтальна табуляція
#10	LF	Переведення рядка
#11	VT	Вертикальна табуляція
#12	FF	Прогін сторінки
#13	CR	Повернення каретки
#26	EOF	Кінець файлу
#27	ESC	Кінець роботи
#32	BL	Пробіл

Для даних символьного типу існує набір процедур та функцій, наведених у табл. 1.10.

Таблиця 1.10 − Функції, які застосовуються для даних символьного типу

Звернення	Призначення	Приклад
ord(S)	Визначає порядковый номер N символу S	ord(‘A’) = 65
chr(N)	Визначає символ S по номеру N	chr(66) = ’A’
pred(S)	Визначає попередній символ S	pred(‘B’) = ’A’
succ(S)	Визначає наступний символ S	succ (‘A’) = ’B’
UpCase(S)	Визначає символ S у верхньому регістрі	UpCase(‘y’) = ’Y’

1.4. Оператор присвоювання

Оператор присвоювання – оператор, який найбільш часто використовується у мові програмування Pascal.

Формат запису:

```
X: = A;
```

де

X − ім'я змінної;

A − вираз;

: = − знак присвоювання.

Робота оператора − у процесі виконання оператора обчислюється значення у правій частині виразу та присвоюється імені змінної.

Тип змінної і тип виразу повинні збігатися, крім випадку, коли вираз відноситься до цілого типу, а змінна до дійсного. При цьому відбувається перетворення обчисленого значення виразу до дійсного типу.

1.5. Умовний оператор

Умовний оператор використовується для реалізації розгалужень у програмах.

Формат запису:

```
If L then OP1 else OP2;
```

де

If – якщо;

then – то;

else – інакше;

L – вираз логичного типу;

OP1, OP2 – оператори.

Робота оператора − Обчислюється значення логічного виразу, якщо обчислене значення істинно, то виконується оператор OP1, в іншому випадку − OP2. Після виконання операторів OP1 або OP2 виконується оператор, наступний за умовним оператором.

Структурна схема базової конструкції наведена на рис. 1.1.

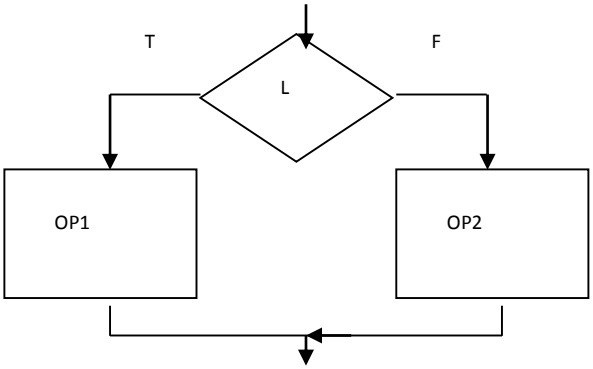


Рисунок 1.1 − Структурна схема умовного оператора

Приклад запису умовного оператора:

```
If  (x<0) and (x>-1) then y:=cos(x) else y:=sin(x);
```

У якості OP1 или OP2 може бути складовий оператор:

```
If x>0 then y:= cos(x)
     begin
       y:= sin(x); x:= x+0.1;
     end;
```

У якості OP1 или OP2 може бути умовний оператор:

```
If  x>0 then y:= cos(x) else if x>0  then y:= sin(x);
```

У мові Pascal допускається коротка форма умовного оператора, яка відповідна структурі, яка наведена на рис. 1.2.

Формат запису:

```
If L then OP;
```

Робота оператора − Обчислюється значення логічного виразу, якщо обчислене значення істинно, то виконується оператор OP, в іншому випадку виконується оператор, наступний за умовним оператором.

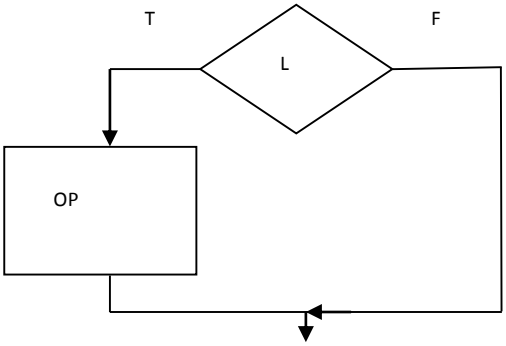


Рисунок 1.2 − Структурна схема умовного оператора

Приклад запису

```
If x>0 then y:=cos(x);
```

Індивідуальні завдання

Похідні дані. Надані у табл. 1.11 та 1.12. Дані обирати згідно з номером за списком у журналі.

Завдання. Ввести з клавіатури **X** та обчислити **A, B** та **Y**. Результат вивести на екран.

Таблиця 1.11 – Варіанти індивідуальних завдань

Перша цифра	0	1	3
Формула	$y = \begin{cases} \frac{A+B}{A \cdot B}, & A < B; \\ \frac{2A-B}{A+1}, & A \geq B \end{cases}$	$y = \begin{cases} \frac{AB+1}{2(A+B)}, & A < B; \\ \frac{A-B}{A+2B}, & A \geq B \end{cases}$	$y = \begin{cases} \frac{A^2-B}{A+B^2}, & A < B; \\ \frac{A-B^2}{A^2+B}, & A \geq B \end{cases}$

Таблиця 1.12 – Варіанти індивідуальних завдань

Друга цифра	Формула	
	A	B
0	e^x	$1-x^2$
1	$\sin x$	$\cos^2 x$
2	x^3	e^x
3	$\cos x$	x^3
4	$1+x^2$	x^3+1
5	e^x	$1+x^2$
6	$\cos x$	e^x
7	x^3+1	x^2
8	x^3+1	e^x
9	e^x	$\sin^2 x$

Приклад виконання лабораторної роботи 1

Індивідуальне завдання. Ввести із клавіатури **X** та обчислити **A, B** та **Y**. Результат вивести на екран.

$$A = e^{x},\ B = \sin^2 x,$$

$$y = \begin{cases} \dfrac{A^2 - B}{A + B^2}, & A < B; \\ \dfrac{A - B^2}{A^2 + B}, & A \geq B. \end{cases}$$

Схема алгоритму:

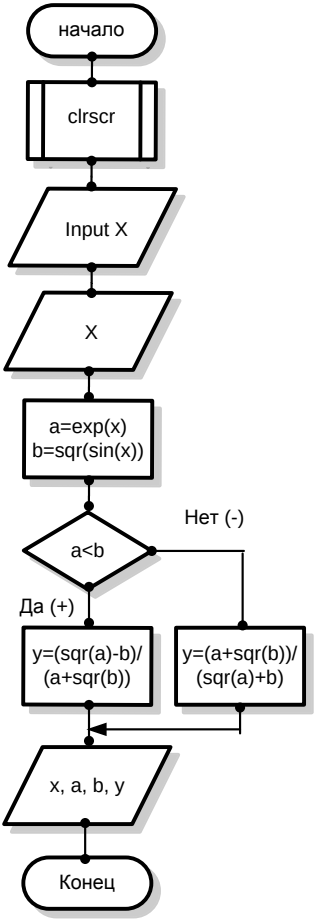


Рисунок 1.3 – Схема алгоритму програми

Текст програми

```
program lab1; {Заголовок программы}
  uses crt; {Подключение стандартного модуля crt}
  var {Раздел описания переменных}
   x,a,b,y : real;
begin {Начало программы}
    clrscr; {Очистка экрана}
{Текстовое сообщение – указание оператору о
дальнейших действиях}
    writeln('Input x');
    readln(x); {Ввод X}
    a:=exp(x); {Вычисление A}
    b:=sqr(sin(x)); {Вычисление B}
{Вычисление Y в зависимости от выполнения или
невыполнения условия a<b}
    if a<b then
      y:=(sqr(a)-b)/(a+sqr(b))
    else
      y:=(a+sqr(b))/(sqr(a)+b);
{Текстовый вывод имен переменных}
    writeln('     X       A       B       Y');
{Вывод результатов}
    writeln(x:8:2,a:8:2,b:8:2,y:8:2);
    readkey; {Ожидание нажатия клавиши}
end. {Конец программы}
```

Отримані результати

При введенні X = 2,5 маємо результат, наведений на рис. 1.4.

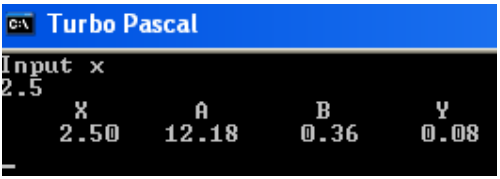


Рисунок 1.4 – Вікно результату запуску програми при X = 2,5

При введенні X = − 4,5 маємо результат, наведений на рис. 1.5.

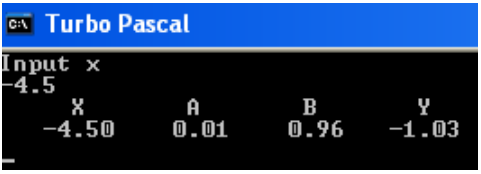


Рисунок 1.5 – Вікно результату запуску програми при X = − 4,5

ЛАБОРАТОРНА РОБОТА 2

ПРОГРАМУВАННЯ ВКЛАДЕНИХ ЦИКЛІВ З ВИКОРИСТАННЯМ МОВИ ПРОГРАМУВАННЯ Pascal

Мета роботи − набуття практичних навичок у розробці та виконанні програм із вкладеними циклами.

Загальні положення

2.1. Оператори циклу

Цикл − це фрагменти програм, які багаторазово повторюються. Алгоритм циклічної структури − це алгоритм, що містить цикли.

У мові програмування Pascal існує три оператори циклу:

– цикл з передумовою;

– цикл з постумовою;

– цикл з параметром.

Для всіх циклів характерні такі особливості:

– значення змінних, що використовуються у циклі та не змінюються в ньому, мають бути визначені до входу в цикл;

– вхід у цикл можливий тільки через його початок;

– вихід із циклу здійснюється як в результаті його природнього закінчення, так і за допомогою операторів переходу.

2.1.1. Оператор циклу з передумовою

Формат запису:

```
While L do OP;
```

де

While − поки не;

do − виконати;

L − вираз логічного типу;

OP − тіло циклу; оператор (простий або складовий).

*Робота оператора. О*бчислюється значення логічного виразу, якщо обчислене значення істинно, то виконується оператор OP, після

чого повторюється перевірка умови і виконання операторів тіла циклу. В іншому випадку здійснюється вихід з циклу.

Оператор циклу з передумовою реалізує таку базову конструкцію:

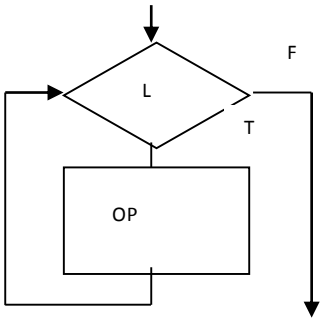


Рисунок 2.1 − Структурна схема оператора цикла з передумовою

Обчислення значення логічного виразу передує виконання операторів тіла циклу, тому цей оператор циклу називається циклом з передумовою.

2.1.2. Оператор циклу з постумовою

Формат запису:

```
Repeat
   OP
Until L;
```

де

Repeat − повторювати;

Until − поки не;

L − вираз логічного типу;

OP − тіло циклу; оператор (простий або складовий).

Робота оператора. Виконується оператор OP, після чого обчислюється значення логічного виразу L, якщо обчислене значення False, то знову виконується оператор OP, в іншому випадку здійснюється вихід з циклу.

Оператор циклу з постумовою реалізує таку конструкцію.

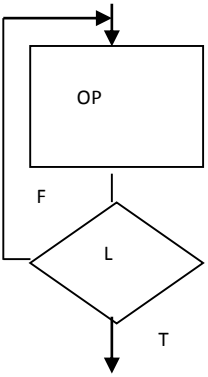


Рисунок 2.2 − Структурна схема оператора циклу з постумовою

Обчислення значення логічного виразу здійснюється після виконання операторів тіла циклу, тому цей оператор циклу називається циклом з постумовою.

На відміну від циклу з передумовою у циклі з постумовою тіло циклу виконується принаймні один раз незалежно від умови.

В операторі циклу з постумовою ключові слова Repeat і Until мають значення операторних дужок.

2.1.3. Оператор циклу з параметром

Формат запису:

```
For P: = Pn to Pk do OP;
```

або

```
For P: = Pk downto Pn do OP;
```

де

For − для;

to − до;

downto − зменшуючи до

do − виконати;

OP − тіло циклу; оператор (простий або складовий);

P − параметр циклу, змінна порядкового типу;

Pn, Pk − початкове і кінцеве значення параметра циклу.

Робота оператора − обчислюється початкове значення параметра циклу Pn і присвоюється параметру P. Перевіряється умова P ≤ Pk, і якщо воно True, то виконуються оператори тіла циклу OP. Після чого нарощується значення P на одиницю і знову перевіряється умова P ≤ Pk. Якщо умова False, то здійснюється вихід з циклу. В операторі з downto крок зміни параметра циклу дорівнює − 1.

Оператор циклу з параметром реалізує таку базову конструкцію.

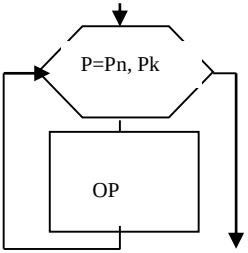


Рисунок 1.3 − Структурна схема оператора циклу з параметром

Індивідуальні завдання

Похідні дані. Надані у таблицях 2.1 та 2.2.

Завдання. Обчислити **y** для усіх комбінацій **x** та **k**. Використовувати оператори циклу **for…**, **while …** або **repeat ….** Видати на екран таблицю значень **k, x, y**.

Таблиця 2.1 – Варіанти індивідуальних завдань

Перша цифра у журналі	0	1	2	3
Діапазон зміни параметрів	$1 \le k \le 3$ $\Delta k = 1$ $0{,}1 \le x \le 0{,}5$ $\Delta x = 0{,}1$	$3 \le k \le 6$ $\Delta k = 1$ $0{,}5 \le x \le 2{,}5$ $\Delta x = 0{,}5$	$6 \le k \le 9$ $\Delta k = 1$ $0{,}2 \le x \le 1{,}0$ $\Delta x = 0{,}2$	$9 \le k \le 12$ $\Delta k = 1$ $0{,}8 \le x \le 1{,}2$ $\Delta x = 0{,}2$

Таблиця 2.2 – Варіанти індивідуальних завдань

Друга цифра у журналі	Функція **y**
0	$y = \frac{k \cdot \sin x + e^{-x}}{1{,}3 \cdot 10^{2} \cdot \sqrt{x}}$
1	$y = \frac{\ln x + x^{k}}{2{,}5 \cdot 10^{1} \cdot \cos^{3} x}$
2	$y = \frac{-0{,}15 \cdot 10 \cdot \sin^{2} x + \cos x^{2}}{e^{-kx}}$
3	$y = \frac{\operatorname{tg} x^{3} + e^{-x}}{3{,}25 \cdot 10^{-1} \cdot \sqrt{x}}$
4	$y = \frac{\sqrt{x} + e^{-kx}}{\cos^{3} x^{2} + 0{,}27 \cdot 10^{-3}}$
5	$y = \frac{\sin^{2} x + \cos x^{3}}{e^{-kx} + 1.35 \cdot 10^{-2}}$
6	$y = \frac{\operatorname{tg}^{2} x + k \cdot \ln x}{\sqrt{x} + 3{,}75 \cdot 10^{-4} \cdot}$
7	$y = \frac{5 \cdot \cos^{2} x + e^{-kx}}{\sqrt{x} + 4{,}28 \cdot 10^{-2} \cdot}$
8	$y = \frac{ln\, x + \sqrt{x}}{k \cdot e^{-3x} + 3{,}89 \cdot 10^{1}}$
9	$y = \frac{k \cdot \operatorname{tg} x + \sqrt{x}}{e^{-kx} + 8{,}13 \cdot 10^{-3}}$

Приклад виконання лабораторної роботи 2

Індивідуальне завдання. Обчислити **y** для всіх комбінацій **x** та **k**. Використовувати оператори циклу **for…**, **while …** або **repeat ….** Видати на екран таблицю значень **k, x, y** для таких умов:

$9 \le k \le 12,\ \Delta k = 1;\ 0{,}8 \le x \le 1{,}2,\ \Delta x = 0{,}2;\ y = \frac{k \cdot \operatorname{tg} x + \sqrt{x}}{e^{-kx} + 8{,}13 \cdot 10^{-3}}$

Алгоритм програми:

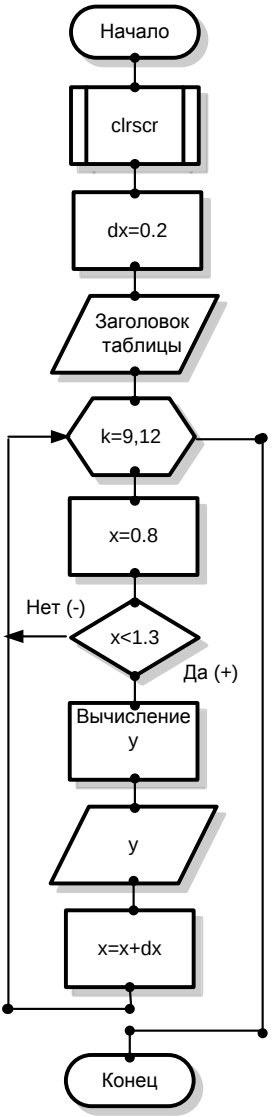


Рисунок 1.4 − Алгоритм програми з використанням циклу з передумовою

Текст програми:

```
{В данной программе для реализации внутреннего циклу
используется оператор циклу с предусловием while…do}
program lab2_1; {Заголовок программы}
  uses crt; {Подключение стандартного модуля crt}
  var {Раздел описания переменных}
   k : integer;
   x,y,dx : real;
begin {Начало программы}
   clrscr; {Очистка экрана}
   dx:=0.2; {Задание шага изменения переменной x}
{Вывод на экран заголовка таблицы}
   writeln('_____________________');
   writeln('   k    x      y');
   writeln('_____________________');
   for k:=9 to 12 do {Заголовок циклу с параметром}
   begin {Начало тела циклу с параметром}
   x:=0.8; {Задание начального значения переменной x}
     while x<1.3 do {Заголовок циклу с предусловием}
      begin {Начало тела циклу с предусловием}
{Вычисление и вывод текущего значения переменной y}
     y:=(k*(sin(x)/cos(x))+sqrt(x))/(exp(x)+8.13e-3);
     writeln(k:4,x:6:2,y:8:2);
     x:=x+dx; {Наращивание значения переменной x}
      end; {Окончание тела циклу с предусловием}
     end; {Окончание тела циклу с параметром}
   readkey; {Ожидание нажатия клавиши}
end. {Конец программы}
```

Алгоритм програми:

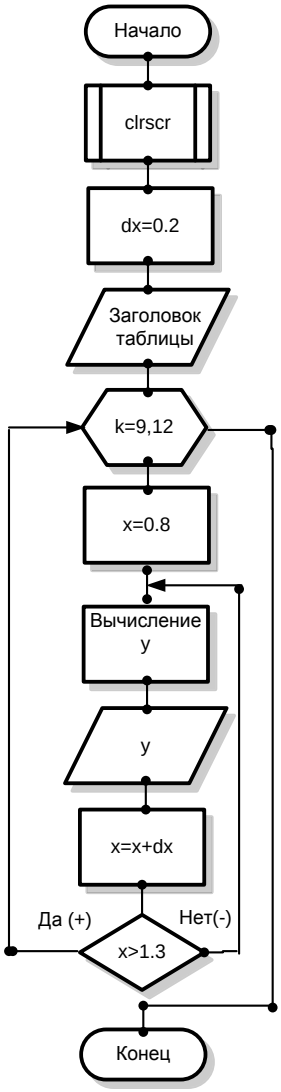


Рисунок 1.5 − Алгоритм програми з використанням циклу з постумовою

Текст програми

```
{В данной программе для реализации внутреннего
циклу используется оператор циклу с постусловием
repeat…unti}
program lab2_2; {Заголовок программы}
  uses crt; {Подключение стандартного модуля crt}
  var {Раздел описания переменных}
   k : integer;
   x,y,dx : real;
begin {Начало программы}
   clrscr; {Очистка экрана}
   dx:=0.2; {Задание шага изменения переменной x}
{Вывод на экран заголовка таблицы}
   writeln('_____________________');
   writeln('   k    x      y');
   writeln('_____________________');
   for k:=9 to 12 do {Заголовок циклу с параметром}
    begin {Начало тела циклу с параметром}
    x:=0.8;{Задание начального значения переменной x}
     repeat {Заголовок циклу с постусловием}
{Вычисление и вывод текущего значения переменной y}
     y:=(k*(sin(x)/cos(x))+sqrt(x))/(exp(x)+8.13e-3);
     writeln(k:4,x:6:2,y:8:2);
       x:=x+dx; {Наращивание значения переменной x}
     until x>1.3; {Окончание циклу с постусловием}
     end; {Окончание тела циклу с параметром}
   readkey; {Ожидание нажатия клавиши}
end. {Конец программы}
```

Отримані результати

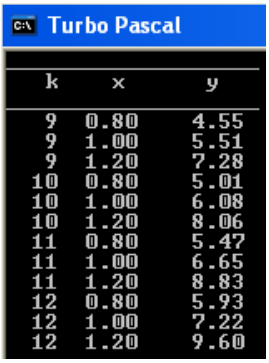


Рисунок 1.6 − Вікно результату запуску програми

ЛАБОРАТОРНА РОБОТА 3

РОБОТА З ФАЙЛАМИ ТА МАСИВАМИ

Мета роботи − набуття практичних навичок в розробці та відлагодженні програм із використанням масивів.

Загальні положення

3.1. Файли

Файл − або іменована область пам'яті на зовнішньому носії інформації (вінчестер, CD-диск та ін.), або логічний пристрій, яким є джерело або приймач інформації (клавіатура, дисплей, принтер, порти та ін.).

Під файлом можна розуміти будь-який набір даних.

Наприклад: програма − вихідна, скомпільова, і т.п. Дані − вхідні, проміжні, результуючі, рисунок, текст, графік, аудіо-відео та ін.).

Характеристики файлів:

- Ім'я;
- розмір;
- атрибути;
- дата і час створення;
- тип.

3.1.1. Імена файлів

Існують такі імена файлів:

1. Файлові змінні − це імена, які використовуються програмою на мові Pascal.

2. Дискові файли − це імена, під якими файли зберігаються на диску і використовуються операційними системами.

3. Імена логічних пристроїв − подають імена стандартних (апаратних) логічних пристроїв.

Імена файлових змінних. Визначаються за правилами Pascal.

Імена дискових файлів. Ім’я містить від одного до восьми дозволених символів, тобто букв, цифр і спеціальних символів: _ - $ # & @! % '^ () {}.

Ім’я починається з будь-якого символу. За ім’ям може прямувати розширення, яке містить від одного до трьох символів, і відокремлюється від імені крапкою. Розширення, як правило, характеризує різновид файлу. У табл. 3.1 наведені приклади розширень деяких файлів.

Таблиця 3.1 − Розширення файлів

Розширення	Файл
1	2
pas	текст програми написаний мовою Pascal
for	текст програми написаний мовою Fortran
bas	текст програми написаний мовою Basic
asm	текст програми написаний мовою Assembler
C	текст програми написаний мовою C
cpp	текст програми написаний мовою C++
obj	об’єктний файл
lib	бібліотечний файл
lst	файл лістинг
bat	пакетний (командний) файл
txt	текстовий файл
arj, rar	архівний файл
dbf	файл база даних
doc	файл документ
drw	файл рисунок
msg	файл повідомлень
sys	системний файл
mnu	файл меню
bak	файл копія
hlp	файл допомога

Продовження табл. 3.1

1	2
html	інтернет файл
xls	файл excel
avi	відео файл
bmp, jpg, gif, tif	графічні файли
com	програмний файл у машинних кодах (програма, що безпосередньо виконується)
exe	програмний файл у машинних кодах (вимагає настройки адреси після завантаження в ОП)

Імена логічних пристроїв. Існують такі імена логічних пристроїв:

CON − консоль (клавіатура / дисплей);

LPT1, LPT2, LPT3 − принтери (паралельні порти);

PRN − синонім LPT1;

COM1 і COM2 − комунікаційні, послідовні порти; AUX синонім COM1;

NUL – зв'язує файл із фіктивним пристроєм. При спробі введення відразу повідомляє про кінець файлу, а при виведенні не виконує ніяких дій (може бути використано при настройці програми);

CLOCK $ − системний годинник;

USB − порти.

3.1.2. Розмір файлу

Розмір файлу (довжина) − це обсяг пам'яті, що займає файл. Мінімальний розмір файлу 1 − запис (128 байт), максимальний − весь диск.

3.1.3. Атрибути файлу

Атрибути файлу визначають спосіб його використання, при цьому може бути встановлено наступні атрибути:

– R (Read-only) − файл тільки для читання (видалити неможна);
– A (Archive) − архівний;

– H (Hidden) − прихований файл (ігнорується багатьма командами DOS);

– S (System) − системний файл.

3.1.4. Дата і час створення файлу

Дата і час створення файлу встановлюється за системним календарем і системним годинником (автономні із вбудованим живленням).

3.1.5 Типи файлів

У мові Турбо-Паскаль існує три типи файлів:

– текстові − складаються із символів, упорядкованих у рядки.

– типізовані – складаються із елементів одного базового типу, який може бути будь-яким, крім файлового.

– нетипізовані − використовуються для доступу до будь-яких файлів, структура яких не має значення. Обмін здійснюється блоками кратними запису (128 байт), або сектора диска (512 байт). У файлі зберігається у символьному вигляді.

3.2. Опис файлів

Опис типів файлових змінних може бути виконано:

1) з використанням розділу типів:

```
Type T1 = File of T4;
T2 = Text;
T4 = File;
Var F1: T1;
F2: T2;
F3: T3;
```

2) безпосередньо у розділі опису змінних:

```
Var F1: File of T4;
F2: Text;
F3: File;
```

де

T1, T2, T3 − імена типу файла;

T4 − тип компонент (базовий);

F1, F2, F3 − імена файлових змінних.

Приклад:

```
Type V = array [1 .. n] of real;
         T1 = File of V;
         T2 = File of Real;
         T3 = text;
Var F1: T1;
        F2: T2;
        F3: T3;
```

3.3. Процедури і функції для роботи з файлами

Всі дії з файлами здійснюються за допомогою процедур та функцій. Для кожного виду файлів існує набір процедур і функцій, які використовуються для роботи з файлами, частина процедур і функцій застосовна до будь-яких видів файлів.

Набір процедур та функцій для роботи з файлами наведено у табл. 3.2.

Таблиця 3.2 − Процедури та функції для роботи з файлами

П/ф	Запис	Призначення	Приклад
1	2	3	4
П	ASSIGN(FP,IF); де: ASSIGN - призначити; FP − файл. змінна;IF − ім’я файлу	Здійснює логічний зв'язок імені дискового файла з ім'ям файлової змінної	`Assign (f1,'D:\TP\BIN\ inp.txt');`

Продовження табл. 3.2

1	2	3	4
П	RESET(FP);	Відкриття файлу для читання (введення)	
П	REWRITE(FP);	Відкриття файлу для запису (виведення). Створюється новий файл або знищується вміст старого	
П	CLOSE(FP);	Закриває файл, але зв'язок з файловою змінною зберігається	
П	RENAME(FP);	Перейменування зв'язаного, але не відкритого файлу	`Assign (f1,'D:\TP\BIN\inp.txt'); Rename (f1, 'D:\TP\BIN\d.txt');`
П	ERASE(FP);	Видалення зв'язаного, але невідкритого файлу	
Ф	EOF(FP);	Визначення кінця файлу (true – якщо досягнуто кінця файлу)	

3.3. Масиви

Масив − це впорядкована, обмежена множина однотипних елементів, об'єднаних загальним ім'ям. Тип компонентів (елементів) називається базовим і може бути будь-яким крім файлового. Для позначення елементів масиву використовується змінна з індексами.

Наприклад: A [25], B [5, 8], X [i], Y [i, j], R [n +1].

Розмір масиву (довжина) - кількість елементів масиву.

Розмірність масиву - кількість індексів в масиві.

Індекс визначає розміщення елемента в масиві. Як індекс може бути використано вираз (індексний), змінна або константа (окремий випадок виразу). Як індекси може бути будь-який порядковий тип крім Longint. Тип індексів повинен бути скалярним, упорядкованим і кінцевим. Найчастіше використовуються індекси інтервального типу, при цьому вони можуть бути від'ємними.

Всі компоненти масиву однаково доступні і можуть вибиратися у довільному порядку.

3.3.1. Опис масівів

У програмі масиви повинні бути описані одним із двох способів:

1) з використанням розділу опису типів:

```
Type T = array [T1] of T2;
Var A, B, C: T;
```

де

T – ім'я типу {регулярний тип};

T1 − тип індексу;

T2 − тип компонентів {базовий тип}.

Тип, визначений за допомогою конструкції `array. .of` називається регулярним.

Наприклад

```
Type
      vec_r = array [1. . N] of real;
      vec1_i = array [1. . N] of integer;
       vec1_l = array [False. . True] of Boolean;
       vec1_S = array ['A'1. . 'Z'] of char;
```

```
Var
       A, B: vec_r;
       N, K: vec_i;
       V, W: vec_l;
       R, S: vec_s;
```

2) без використання розділу опису типів:

```
Var  A, B, C  : = array[T1] of T2;
```

Наприклад

```
Var
       A, B: array [1. . N] of real;
       N, K: array [1. . N] of integer;
       V, W: array [False. . True] of Boolean;
       R, S: array ['A'1. . 'Z'] of char;
```

Перший спосіб є кращим, хоча він і подовжує розмір описів, але програма стає більш ясною логічно.

До компонентів масиву можуть бути застосовні операції, процедури і функції, допустимі для даного базового типу.

Всі дії з масивами виконуються в циклі, за винятком привласнення цілком елементів одного масиву іншому.

Наприклад: X: = Y.

3.3.2. Введення / виведення масивів

Оператори введення/виведення виконуються у циклі, при цьому в список введення/виведення поміщається змінна з індексом. Вводити дані можна у пакетному або діалоговому (інтерактивному) режимі. Наприклад,

1)
```
For i: = 1 to n do Read (x [i]);
  Readln;
```

На екрані:
– 2.1 3.8. . . 5.7

2) `For i: = 1 to n do Readln (x [i]);`

На екрані:
– 2.1
3.8
. . .
5.7

Елементи масиву при введенні рядка повинні бути розділені довільною кількістю пробілів, в кінці рядка ставиться повернення каретки.

Аналогічно можна записати процедури виведення (див. приклад), але при цьому необхідно передбачити розділові пробіли і формати виведених даних.

Індивідуальні завдання

Похідні дані. Надані у таблицях 3.1 та 3.2.

Завдання. Скласти програму згідно з умовами в табл. 3.2.

Таблиця 3.3 – Варіанти індивідуальних завдань

Перша цифра у журналі	Масиви	Довжина
0	1,3; 0; 5,8; 4,03; – 21,5; 7; 0,02	7
1	0,5; 1,2; – 3,8; – 0,99; 0,1; 1,7; 13; – 4	8
2	15; 0; – 31,2; 47,8; 3; – 17,1; 0,04; 0,3	8

Таблиця 3.4 – Варіанти індивідуальних завдань

Др.цифра у журналі	Умова задачі
0	Визначити суму та кількість додатних елементів масиву, а також добуток та кількість від'ємних елементів масиву
1	Визначити максимальний елемент із від'ємних чисел, а також його порядковий номер
2	Визначити добуток та кількість елементів масиву менших ніж 1 та рівних нулю
3	Визначити добуток максимального та мінімального елементів масиву
4	Визначити $y = \begin{cases} \dfrac{a_i^3 + \sqrt{\lvert a_i + 1 \rvert}}{\cos^2 a_i}, & якщо\ a_i \le 2; \\ a_i^{5/7} + e^{-5a_i}, & якщо\ a_i < 2 \end{cases}$
5	Визначити добуток та кількість елементів масиву, які не дорівнюють нулю, а також суму та кількість від'ємних елементів масиву
6	Для елементів масиву з парними індексами визначити максимальний елемент, а також його порядковий номер
7	Визначити сумму максимального та мінімального елементів масиву
8	Для елементів масиву з непарними індексами визначити мінімальний елемент із від'ємних чисел, а також його порядковий номер
9	Визначити суму та кількість додатних елементів масиву, а також суму та кількість від'ємних елементів масиву

Приклад виконання лабораторної роботи 3

Індивідуальне завдання

Для масиву, що складається з 8 елементів: 15; 0; − 31,2; 47,8; 3; − 17,1; 0,04; 0,3, визначити суму та кількість додатних елементів масиву, а також суму та кількість від'ємних елементів масиву.

Схема алгоритму:

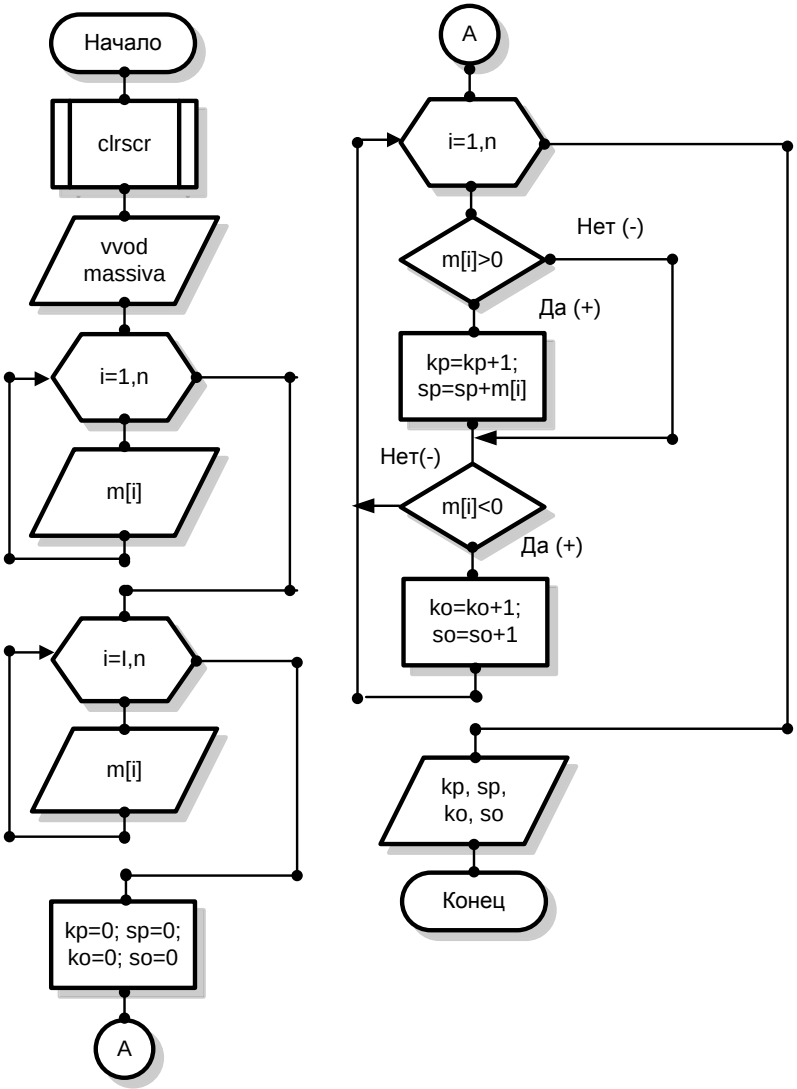


Рисунок 3.1 – Схема алгоритму програми

Текст програми

```
program lab3_1; {Заголовок программы}
 uses crt; {Подключение стандартного модуля crt}
 const {Раздел описания констант}
  n=8; {Количество элементов массива}
 type {Раздел описания пользовательских типов данных}
  mas=array[1..n] of real;
var {Раздел описания переменных}
  m : mas; {m-массив}
{ kp,ko-количество положительных и отрицательных
элементов массива соответственно, i-параметр циклу}
  kp,ko,i : integer;
{ sp,so-сумма положительных и отрицательных элементов
массива соответственно}
  sp,so : real;
begin {Начало программы}
  clrscr; {Очистка экрана}
{Указание оператору о его дальнейших действиях}
  writeln('Vvesti ',n,' elementov massiva');
   for i:=1 to n do
    readln(m[i]); {Ввод с клавиатуры элементов массива}
  writeln('                    Massiv:');
   for i:=1 to n do
    write(m[i]:8:2); {Вывод на экран элементов массива}
   writeln;
{Обнуление переменных kp, ko, sp, so}
   kp:=0;
   ko:=0;
   sp:=0;
   so:=0;
   for i:=1 to n do
    begin {Начало тела циклу с параметром}
```

```
      if m[i]>0 then
       begin
{Вычисление текущих значений kp и sp}
        kp:=kp+1;
        sp:=sp+m[i];
       end;
      if m[i]<0 then
       begin
{Вычисление текущих значений ko и so}
        ko:=ko+1;
        so:=so+m[i];
       end;
    end; {Конец тела циклу с параметром}
{Вывод результатов}
  writeln('k-vo pol.=',kp:4,' summa pol.=',sp:6:2);
  writeln('k-vo otr.=',ko:4,' summa otr.=',so:6:2);
 readkey; {Ожидание нажатия клавиши}
end. {Конец программы}
```

Отримані результати

```
Turbo Pascal
Vvesti 8 elementov massiva
15
0
-31.2
47.8
3
-17.1
0.04
0.3
                Massiv:
   15.00    0.00  -31.20   47.80    3.00  -17.10    0.04    0.30
k-vo pol.=   5 summa pol.= 66.14
k-vo otr.=   2 summa otr.=-48.30
```

Рисунок 3.2 - Вікно результату запуску програми

Схема алгоритму

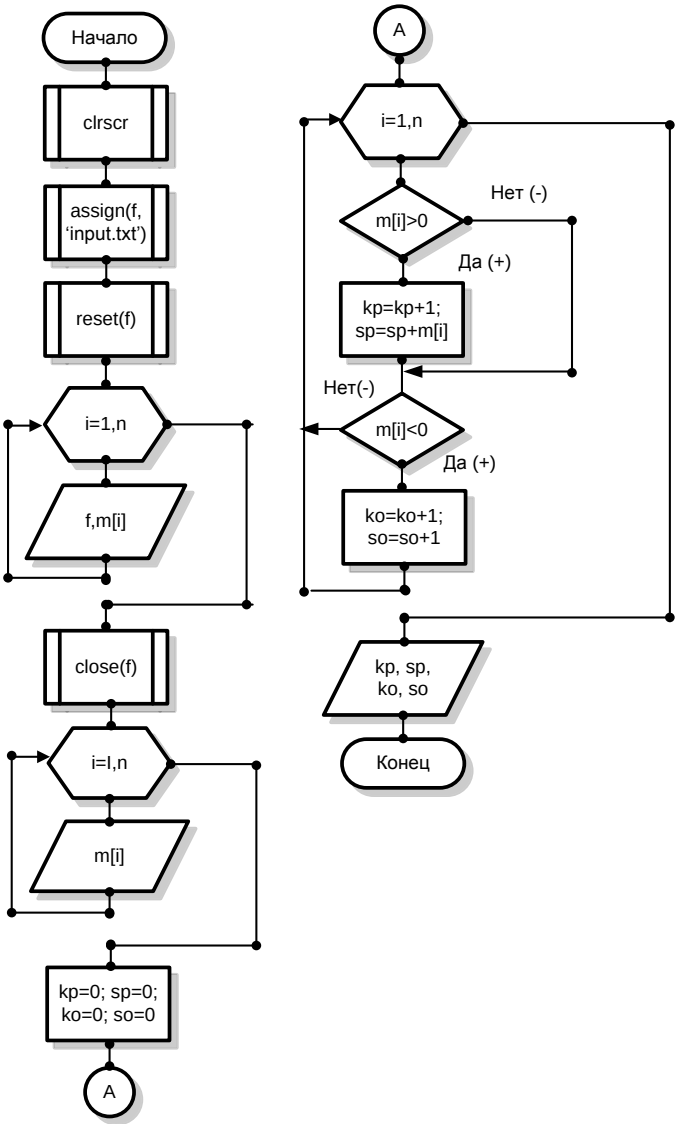


Рисунок 3.3 – Схема алгоритму програми

Текст програми

```
program lab3_2; {Заголовок программы}
 uses crt; {Подключение стандартного модуля crt}
 const {Раздел описания констант}
  n=8; {Количество элементов массива}
 type {Раздел описания пользовательских типов данных}
  mas=array[1..n] of real;
 var {Раздел описания переменных}
  m : mas; {m-массив}
  f : text; {f-файловая переменная}
{ kp,ko-количество положительных и отрицательных
элементов массива соответственно, i-параметр циклу}
  kp,ko,i : integer;
{ sp,so-сумма положительных и отрицательных элементов
массива соответственно}
  sp,so : real;
begin {Начало программы}
  clrscr; {Очистка экрана}
{Связывание файловой переменной f с файлом input.txt}
  assign(f,'input.txt');
  reset(f); {Открытие файла для чтения}
   for i:=1 to n do
    read(f,m[i]); {Чтение данных из файла в массив}
  close(f); {Закрытие файла}
  writeln('                    Massiv:');
   for i:=1 to n do
    write(m[i]:8:2); {Вывод на экран элементов массива}
   writeln;
{Обнуление переменных kp, ko, sp, so}
   kp:=0;
   ko:=0;
   sp:=0;
   so:=0;
   for i:=1 to n do
    begin {Начало тела циклу с параметром}
     if m[i]>0 then
```

```
      begin
{Вычисление текущих значений kp и sp}
       kp:=kp+1;
       sp:=sp+m[i];
      end;
     if m[i]<0 then
      begin
{Вычисление текущих значений ko и so}
       ko:=ko+1;
       so:=so+m[i];
      end;
    end; {Конец тела циклу с параметром}
{Вывод результатов}
  writeln('k-vo pol.=',kp:4,' summa pol.=',sp:6:2);
  writeln('k-vo otr.=',ko:4,' summa otr.=',so:6:2);
 readkey; {Ожидание нажатия клавиши}
end. {Конец программы}
```

Вміст вхідного файлу input.txt

Рисунок 3.4 – Вікно із вмістом вхідного файлу

Отримані результати

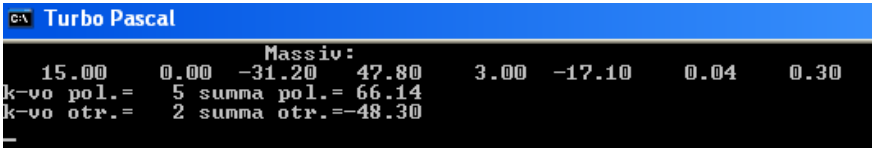


Рисунок 3.5 – Вікно результату запуску програми

ЛАБОРАТОРНА РОБОТА 4

РОБОТА З МАТРИЦЯМИ

Мета роботи – набуття практичних навичок в розробці та відлагодженні програм з використанням матриць.

Загальні положення

4.1. Багатовимірні масиви

У мові програмування Pascal ніяких обмежень на тип компонентів (крім файлового) не накладається, потрібно, щоб вони були одного базового типу. Компонентами масиву можуть бути масиви. Число вкладень, тобто розмірність, може бути будь-якою, але при цьому сумарна довжина внутрішнього подання повинна бути не більше 65 535 байт.

В залежності від розмірності розрізняють:

- одновимірні масиви − вектори;
- двовимірні масиви − матриці;
- тривимірні масиви;
- та ін.

З багатовимірних масивів найбільш часто використовуються двовимірні масиви − матриці. Вони зручні для обробки таблиць та інших структур.

4.1.1. Матриці

Компоненти матриці позначаються змінною з двома індексами.

Наприклад: A [i, j] або A [i], [j], при цьому перший індекс позначає номер рядка, а другий − номер стовпця.

Матриці можуть бути описані з використанням розділу типів або безпосередньо у розділі опису змінних. Є кілька способів опису матриць.

1) з використанням розділу опису типів:

```
a) Type M = array [T1, T2] of T3;
b) Type M = array [T1] of array [T2] of T3;
```

```
c) Type V = array [T1] of T3;
      Type M = array [T2] of V;
```

де

V – імя' типу вектор;

M – ім'я типу матриця;

T1, T2 – тип індексів (м.б. різними);

T3 – тип компонентів.

Наприклад

```
a) Type M = array [1 .. n, 1 .. m] of real;
   Vec A, B: M;
b) Type M = array [1 .. n] of array [1 .. m] of real;
   Vec A, B: M;
c) Type V = array [1 .. n] of real;
   Type M = array [1 .. m] of V;
   Vec A, B: M;
```

Третій спосіб доцільно використовувати, якщо необхідно працювати з окремими рядками або стовпцями матриці;

2) без використання розділу опису типів:

```
Var A, B: array [T1, T2] of T3;
```

Наприклад:

```
Var A, B: array [1 .. n, 1 .. m] of real;
```

Як зазначалося раніше, використання розділу Type свідчить про добрий стиль програмування.

Всі дії з матрицями виконуються з використанням подвійних циклів (за рядками і за стовпцями), за винятком привласнення цілком елементів однієї матриці інший. Наприклад: X: = Y.

4.1.2. Введення/виведення матриць

Введення/виведення матриць виконуються з використанням вкладених циклів, при цьому в список введення/виведення поміщається змінна з індексами.

Вводити матриці можна за елементами або за рядками.

1)
```
For i: = 1 to n do
   For j: = 1 to n do Read (x [i, j]);
```

На екрані:

– 2
3
. . .
5

2)
```
For i: = 1 to n do
  Begin
    For j: = 1 to n do Read (x [i, j]);
    Readln;
  End;
```

На екрані:

– 2 3 …5
. . .

Аналогічно можна записати процедури виведення, але при цьому необхідно передбачити розділові пробіли і формати виведення даних.

```
For i: = 1 to n do
     Begin
        For j: = 1 to n do Write (x [i, j]: 4);
        Writeln;
     End;
```

Індивідуальні завдання

Похідні дані. Надані у табл. 4.1.

Завдання. Скласти програму згідно з умовами в табл. 4.1.

Таблиця 4.1 – Варіанти індівідуальних завдань

1	N x m	R/I	Умова задачі
0	5 x 3	Ціл.	Для кожного рядка визначити максимальний елемент масиву, а також номер його стовпця
1	4 x 5	Дійс.	Для кожного стовпця визначити суму додатних кількість від'ємних елементів масиву
2	3 x 2	Ціл.	Для кожного рядка визначити добуток та кількість від'ємних елементів масиву
3	3 x 4	Дійс.	Для кожного стовпця визначити мінімальний елемент масиву а також номер його рядка
4	4 x 4	Ціл.	Для кожного рядка визначити суму та кількість від'ємних елементів масиву
5	5 x 3	Дійс.	Для кожного стовпця визначити добуток та кількість додатних елементів масиву
6	3 x 3	Ціл.	Для кожного рядка визначити суму та кількість елементів масиву більших ніж 2
7	4 x 5	Дійс.	Для кожного стовпця визначити добуток елементів не рівних 0 та суму додатних елементів масиву
8	3 x 2	Ціл.	Для кожного рядка визначити суму від'ємних елементів масиву та знайти максимальний елемент
9	3 x 5	Дійс.	Для кожного стовпця визначити добуток додатних елементів масиву та знайти мінімальний елемент

Матрицю задати довільно з урахуванням другого і третього стовпців із табл. 4.1 та розташувати у файлі **inp.txt.**

Приклад виконання лабораторної роботи 4

Індивідуальне завдання. Для матриці, що містить три рядки та п'ять стовпців дійсних елементів, визначити суму додатних елементів та мінімальний елемент у кожному стовпці. Матрицю задати довільно та розташувати у текстовому файлі **inp.txt.**

Схема алгоритму

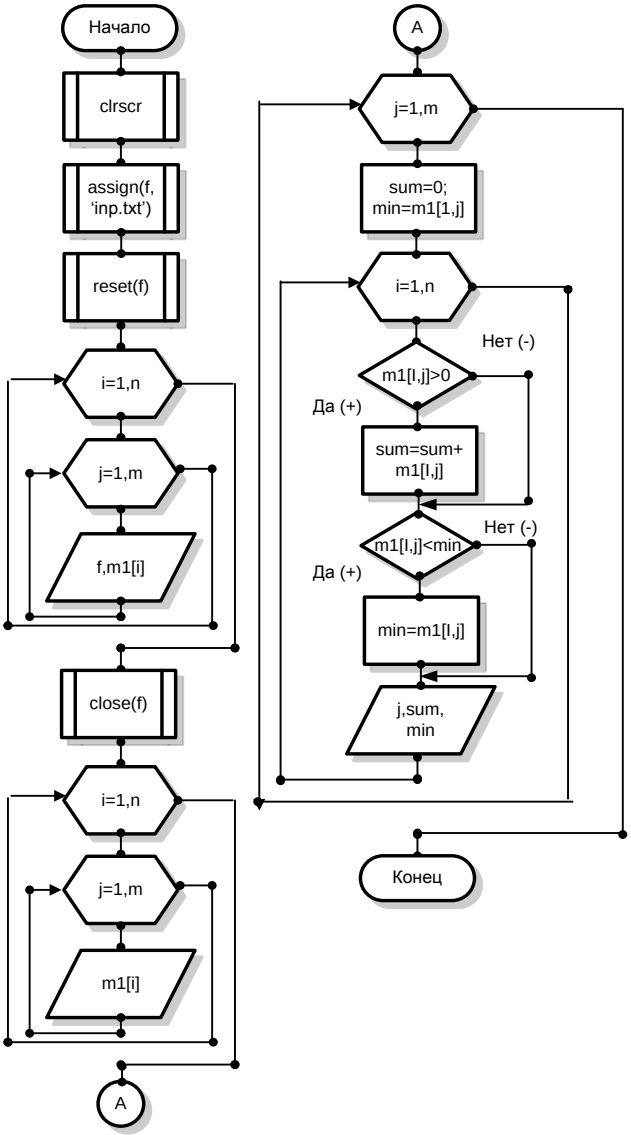


Рисунок 4.1 – Алгоритм програми

Текст програми

```
program lab4; {Заголовок программы}
uses crt; {Подключение стандартного модуля crt}
 const {Раздел описания констант}
  n=3; {Количество строк матрицы}
  m=5; {Количество столбцов матрицы}
 type {Раздел описания пользовательских типов данных}
  matr=array[1..n,1..m] of real;
 var {Раздел описания переменных}
  m1 : matr; {m1-матрица}
  i,j : integer; {Параметры циклов}
  f : text; {f-файловая переменная}
{sum-сумма положительных элементов в столбце, min-
минимальный элемент в столбце}
  sum,min : real;
begin {Начало программы}
 clrscr; {Очистка экрана}
{Связывание файловой переменной f с файлом inp.txt}
 assign(f,'inp.txt');
 reset(f); {Открытие файла для чтения}
 for i:=1 to n do
   for j:=1 to m do
    read(f,m1[i,j]); {Чтение данных из файла в матрицу}
close(f); {Закрытие файла}
 writeln('              Matrica:');
 for i:=1 to n do
  begin
   for j:=1 to m do
{Вывод на экран элементов матрицы}
    write(m1[i,j]:8:2);
    writeln;
  end;
    writeln;
{Просмотр матрицы по столбцам}
   for j:=1 to m do
    begin {Начало тела внешнего циклу (по столбцам)}
```

```
{Обнуление суммы положительных элементов по текущему
столбцу. Присвоение минимальному элементу значения из
первой строки текущего столбца.}
     sum:=0;
     min:=m1[1,j];
      for i:=1 to n do
{Начало тела внутреннего циклу (по строкам)}
       begin
{Вычисление суммы и поиск min по столбцу при выполнении
соответствующих условий}
             if m1[i,j]>0 then sum:=sum+m1[i,j];
        if m1[i,j]<min then min:=m1[i,j];
       end; {Конец тела внутреннего циклу (по строкам)}
{Вывод значения суммы положительных элементов и
минимального значения для текущего столбца}
      writeln('stolbec# ',j,' sum=',sum:6:2,'
min=',min:6:2);
    end; {Конец тела внешнего циклу (по столбцам)}
 readkey; {Ожидание нажатия клавиши}
end. {Конец программы}
```

Вміст вхідного файлу inp.txt

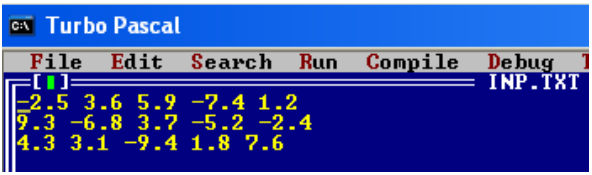


Рисунок 4.2 – Вікно із вмістом вхідного файлу **inp.txt**

Отримані результати

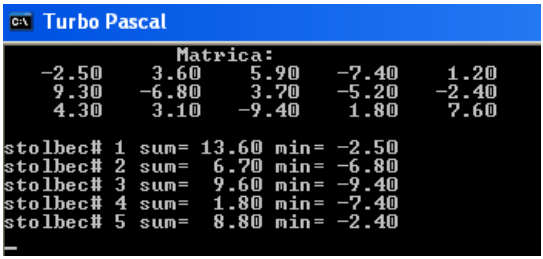


Рисунок 4.3 – Вікно результату запуску програми

ЛАБОРАТОРНА РОБОТА 5

ПРОЦЕДУРИ ТА ФУНКЦІЇ

Мета роботи − набуття практичних навичок у розробці та відлагодженні програм з використанням процедур та функцій.

Загальні положення

5.1. Підпрограми

5.1.1. Процедури та функції

Підпрограма - самостійний фрагмент програми, що реалізує певний алгоритм і допускає багаторазове звернення до нього із різних частин програми.

У мові Паскаль існує великий набір стандартних (бібліотечних) процедур і функцій. Процедури і функції містяться у стандартних модулях і підключаються до будь-якої програми або за допомогою ключового слова Uses (наприклад, CRT, GRAPH), або автоматично (наприклад, System).

Принципи хорошого стилю програмування вимагають широкого використання власних підпрограм, які складаються та використовуються також як і бібліотечні.

Застосування підпрограм дозволяє використовувати сучасні технології конструювання програм: структурне, модульне, об’єктне. При цьому складна задача розбивається на ряд підзадач, для кожної з яких складається набір підпрограм, з яких компонується загальна програма.

Мова Pascal містить два типи підпрограм:

- процедури;
- функції.

5.2 Процедури

Структура процедури аналогічна структурі програми і складається із заголовка і блока (тіла процедури).

```
PROCEDURE <ім'я> (<сп. Форм. пар.>);
<Блок>
```

де

PROCEDURE − зарезервоване слово процедура;

<Ім'я> − ім’я процедури, є унікальним, обирається за загальними правилами, бажано, щоб воно відображало сенс процедури;

<Сп. форм. пар.> − список формальних параметрів тобто список імен, що позначають вхідні дані та результат роботи процедури із зазначенням їх типів;

<Блок> − тіло процедури вміщує розділи описів і розділ операторів, що являє собою складовий оператор (сукупність операторів, вкладених в операторні дужки BEGIN END).

Розділи описів процедури містять ті ж розділи, що й основна програма, у тому числі описи підпрограм нижчого рівня (вкладених).

Оператор виклику процедури активізує процедуру.

```
<Ім’я> (<сп. Факт. Пар.>);
```

де:

<Ім'я> − ім’я процедури;

<Сп. факт. пар.> − список фактичних параметрів.

Це список конкретних значень, імен і висловів, що підставляються замість формальних параметрів, і переданих у підпрограму, а також повертаються результати обробки.

Список фактичних параметрів може бути відсутнім.

Між формальними і фактичними параметрами існує взаємо-однозначна відповідність за кількістю, порядком входження і типом.

5.3. Функції

Структура опису та механізм використання функції аналогічний процедурі з урахуванням деяких особливостей. Опис складається із заголовка і блока (тіла функції).

```
FUNCTION <ім'я> (<сп. Форм. Пар.>): <Тип>;
<Блок>;
```

де

FUNCTION − зарезервоване слово функція;

<Ім'я> − ім’я функції, є унікальним, обирається за загальними правилами, бажано, щоб воно відображало сенс функції;

<Сп. форм. пар.> − список формальних параметрів, тобто список імен, що позначають вхідні дані функції із зазначенням їх типів;

<Тип> - тип результату, що повертається функцією і присвоюється імені функції;

<Блок> − тіло процедури подає розділи описів і розділ операторів, що показує складовий оператор (сукупність операторів, вкладених в операторні дужки BEGIN END).

Розділи описів функції містять ті ж розділи, що й основна програма, у тому числі описи підпрограм нижчого рівня (вкладених).

У змістовній частині функції її імені повинен бути присвоєний результат роботи функції.

Виклик функції є операнд (покажчик функції) на відміну від процедури звернення, до якої показує оператор.

Він має вигляд:

```
<Ім'я> (<сп. Факт. Пар.>);
```

де:

<Ім'я> – ім’я функції (покажчик);

<Сп. факт. пар.> – список фактичних параметрів.

Це список конкретних значень, імен і висловів, що підставляються замість формальних параметрів та передаються у функцію.

5.4. Відмінність процедури від функції

Процедури і функції мають такі відмінності:

– функція на відміну від процедури повертає єдине скалярне значення;

– результат обчислення функції присвоюється імені, а процедури входить до списку параметрів;

– ім’я функції має тип;

– звернення до функції являє собою операнд, а процедури – оператор.

Індивідуальне завдання

Похідні дані. Надані у табл. 5.1.

Завдання.: У файл **inp.txt** записати три довільні масиви чисел, тип та довжина яких задана у таблиці. З використанням процедур та функцій скласти програму згідно з умовами в табл. 5.1.

Таблиця 5.1 – Варіанти індивідуальних завдань

№	I/ R	n1, n2, n3	З використанням функції визначити	З використанням процедури визначити
0	R	5, 4, 3	Максимальний елемент масивів	Суму та кількість додатних елементів масивів
1	I	4, 5, 6	Добуток додатних елементів масивів	Максимальний елемент із від'ємних чисел, а також його порядковий номер
2	R	6, 5, 4	Суму додатних елементів масивів	Добуток та кількість елементів масивів менших ніж один та рівних нулю
3	I	5, 4, 6	Мінімальний елемент із від'ємних чисел масивів	Середнє арифметичне двох найменших елементів масивів
4	R	4, 3, 6	Максимальний елемент із від'ємних чисел масивів	Суму від'ємних елементів масивів та їх кількість
5	I	5, 6, 3	Мінімальний елемент масивів	Добуток та кількість елементів масивів не рівних нулю а також суму та кількість від'ємних елементів масивів
6	R	6, 5, 7	Максимальний елемент із додатних чисел масивів	Для елементів масивів з парними індексами максимальний елемент, а також його порядковий номер
7	I	6, 8, 3	Добуток від'ємних елементів масивів	Середнє геометричне двох найбільших елементів масивів
8	R	8, 4, 5	Суму від'ємних елементів масивів	Для елементів масивів з непарними індексами мінімальний елемент із від'ємних чисел, а також його порядковий номер
9	R	9, 10, 7	Максимальний елемент із парних елементів масивів	Суму та кількість додатних елементів масивів

Приклад виконання лабораторної роботи 5

Індивідуальне завдання. У файл **inp.txt** записати три довільні масиви дійсних чисел. Кількість елементів у масивах 9, 6, 7. З використанням функції у кожному з масивів визначити максимальний елемент серед елементів з парними індексами. З використанням процедури у кожному з масивів визначити кількість та суму додатних і від'ємних елементів.

Схеми алгоритмів

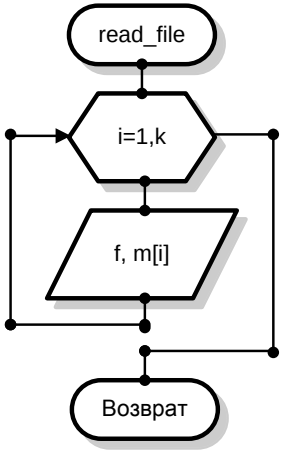


Рисунок 5.1 − Схема алгоритму процедури **read_file**

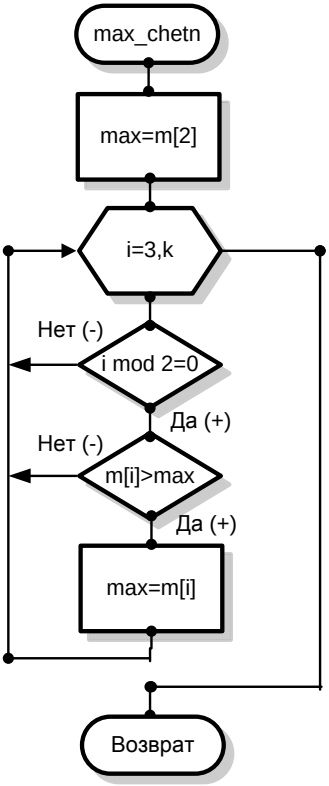


Рисунок 5.2 − Схема алгоритму функції max_chetn

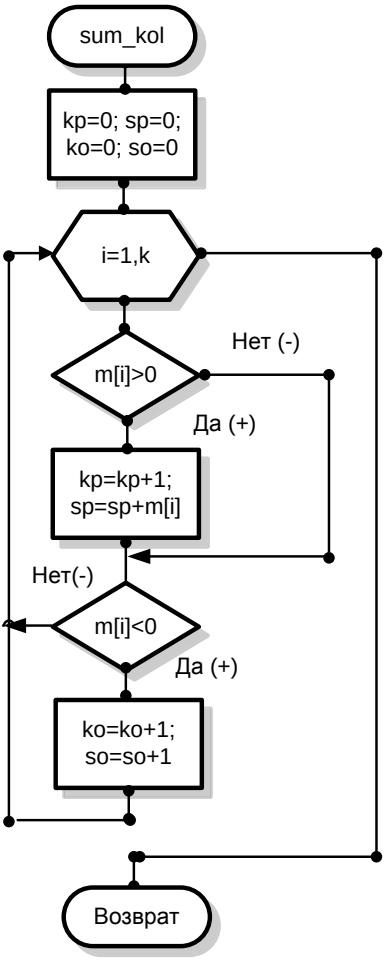


Рисунок 5.3 − Схема алгоритму процедури **sum_kol**

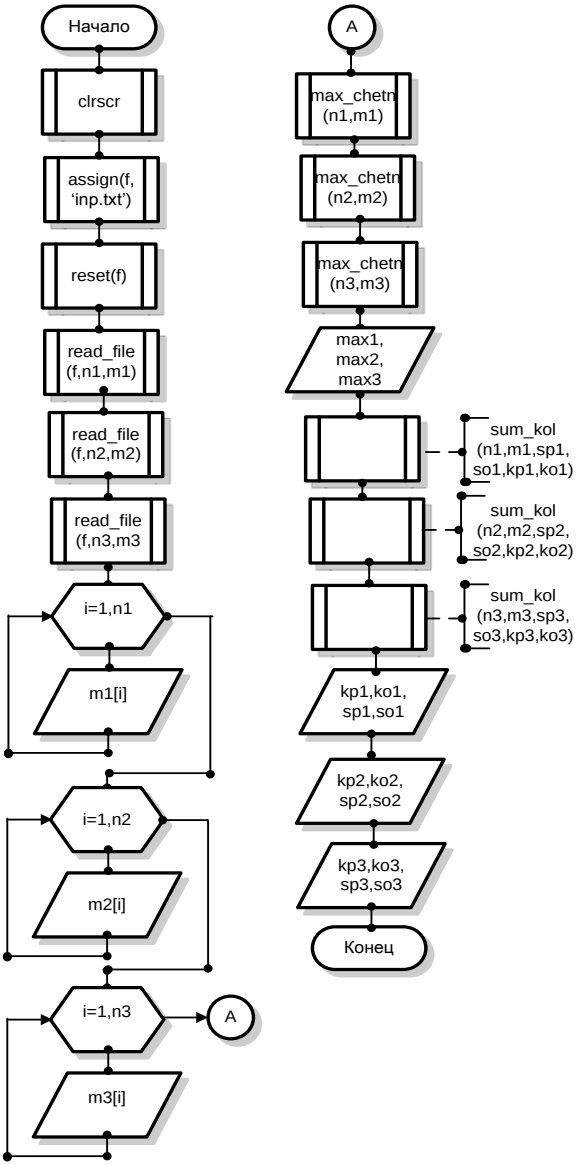


Рисунок 5.4 − Схема алгоритму головної програми

Текст програми

```
program lr5; {Заголовок программы}
 uses crt; {Подключение стандартного модуля crt}
 const {Раздел описания констант}
  n1=9; {Количество элементов в первом массиве}
  n2=6; {Количество элементов во втором массиве}
  n3=7; {Количество элементов в третьем массиве}
 type {Раздел описания пользовательских типов
данных}
{Количество элементов выбирается максимальным из
трех}
  mas=array[1..n1] of real;
{Процедура чтения из файла inp.txt в массив. В
процедуру передаются файловая переменная f и количество
элементов в массиве k. Процедура возвращает массив m,
считанный из текстового файла.}
procedure read_file(var f:text; k:integer; var
m:mas);
  var {Раздел описания локальных переменных}
   i : integer; {Параметр циклу}
begin {Начало процедуры}
{Чтение k элементов из файла и запись их в массив}
   for i:=1 to k do
    read(f,m[i]);
 end; {Конец процедуры}
{Функция определяет максимальный из элементов с
четными индексами. В функцию передаются количество
элементов в массиве k и массив m. Функция возвращает
значение типа real.}
function max_chetn(k:integer; m:mas):real;
 var {Раздел описания локальных переменных}
  max : real; {Вспомогательная переменная}
  i : integer; {Параметр циклу}
begin {Начало функции}
{Выбирается первый элемент с четным индексом}
  max:=m[2];
```

```
     {Просмотр массива, начиная с третьего индекса до
конца. Проверка индекса на четность и сравнение элемента
с четным индексом с текущим значением max.}
         for i:=3 to k do
          if (i mod 2 = 0) and (m[i]>max) then
           max:=m[i];
     {Присвоение имени функции вычисленного значения
max.}
        max_chetn:=max;
     end; {Конец функции}
     {Процедура вычисляет сумму и количество
положительных и отрицательных элементов в массиве. В
процедуру передаются количество элементов в массиве k и
массив m. Процедура возвращает суммы положительных sp и
отрицательных so элементов и количества этих элементов
kp и ko соответственно}
     procedure sum_kol(k:integer; m:mas; var sp,
so:real; var kp, ko:integer);
      var {Раздел описания локальных переменных}
       i : integer; {Параметр циклу}
     begin {Начало процедуры}
     {Обнуление сумм и количеств положительных и
отрицательных элементов}
        sp:=0;
        so:=0;
        kp:=0;
        ko:=0;
         for i:=1 to k do
          begin {Начало тела циклу с параметром}
           if m[i]>0 then
            begin
     {Подсчет количества и суммы положительных
элементов}
             kp:=kp+1;
             sp:=sp+m[i];
            end;
          if m[i]<0 then
```

```
      {Подсчет количества и суммы отрицательных
элементов}
            begin
             ko:=ko+1;
             so:=so+m[i];
            end;
          end; {Конец тела циклу с параметром}
      end; {Конец процедуры}
      {Главная программа}
       Var {Раздел описания локальных переменных}
       m1,m2,m3 : mas; {Массивы}
      {Максимальные значения и суммы положительных и
отрицательных элементов для трех массивов.}
       max1,max2,max3,sp1,so1,sp2,so2,sp3,so3 : real;
      {Параметр циклу i и количества положительных и
отрицательных элементов для трех массивов.}
       i,kp1,ko1,kp2,ko2,kp3,ko3 : integer;
       f : text; {f-файловая переменная}
      begin{Начало программы}
       clrscr; {Очистка экрана}
      {Связывание файловой переменной f с файлом inp.txt}
       assign(f,'inp.txt');
       reset(f); {Открытие файла для чтения}
      {3 раза вызывается процедура read_file для чтения
из файла и записи в массивы m1, m2 и m3 соответственно}
       read_file(f,n1,m1);
       read_file(f,n2,m2);
       read_file(f,n3,m3);
       close(f); {Закрытие файла}
      {Вывод массивов m1, m2 и m3 }
       writeln('                    massiv m1:');
        for i:=1 to n1 do
         write(m1[i]:8:2);
         writeln;
       writeln('                   massiv m2:');
        for i:=1 to n2 do
         write(m2[i]:8:2);
         writeln;
```

```
      writeln('                    massiv m3:');
       for i:=1 to n3 do
        write(m3[i]:8:2);
        writeln;
     {3 раза вызывается функция max_chetn для
определения максимальных элементов, стоящих на четных
местах,  в массивах m1, m2 и m3. Значения, полученные с
помощью функции присваиваются переменным max1, max2 и
max3 соответственно.}
       max1:=max_chetn(n1,m1);
       max2:=max_chetn(n2,m2);
       max3:=max_chetn(n3,m3);
      writeln;
     {Вывд значений переменных max1, max2 и max3}
      writeln('m1: max element s chetnym
indeksom=',max1:6:2);
      writeln('m2: max element s chetnym
indeksom=',max2:6:2);
      writeln('m3: max element s chetnym
indeksom=',max3:6:2);
      writeln;
     {Три раза вызывается процедура sum_kol для подсчета
сумм и количеств положительных и отрицательных элементов
в массивах m1, m2 и m3. Полученные в процедуре значения
возвращаются в переменные: sp1, so1, kp1, ko1 для
массива m1; sp2, so2, kp2, ko2 для массива m2; sp3, so3,
kp3, ko3 для массива m3.}
      sum_kol(n1,m1,sp1,so1,kp1,ko1);
      sum_kol(n2,m2,sp2,so2,kp2,ko2);
      sum_kol(n3,m3,sp3,so3,kp3,ko3);
     {Вывод сумм и количеств положительных и
отрицательных элементов в массивах m1, m2 и m3.}
      writeln('Massiv  Kol+   Sum+   Kol-  Sum-');
      writeln('  m1',kp1:6,sp1:10:2,ko1:4,so1:10:2);
      writeln('  m2',kp2:6,sp2:10:2,ko2:4,so2:10:2);
      writeln('  m3',kp3:6,sp3:10:2,ko3:4,so3:10:2);
      readkey; {Ожидание нажатия клавиши}
     end. {Конец главной программы}
```

Вміст вхідного файлу inp.txt

Рисунок 5.5 – Вікно із вмістом вхідного файлу **inp.txt**

Отримані результати

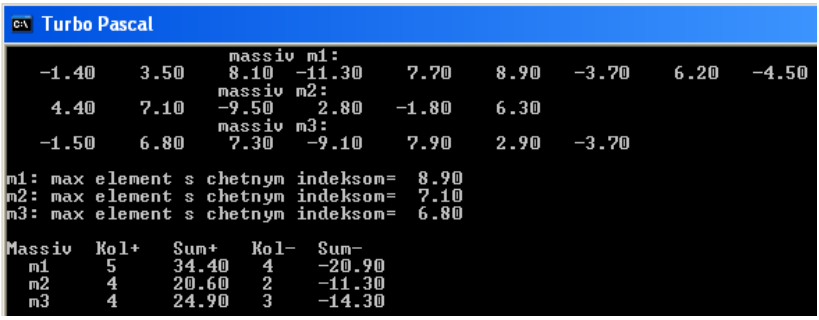


Рисунок 5.6 – Вікно результату запуску програми

ЛАБОРАТОРНА РОБОТА 6

ВЛАСНІ МОДУЛІ В ТР

Мета роботи − набуття практичних навичок у розробці та відлагодженні програм з використанням власних модулів.

Загальні положення

6.1 Власні модулі

6.1.1. Модульне програмування

Існує два підходи до розробки складних програм:

− програмування «зверху вниз», при цьому складається програма, а потім окремі її частини оформляються у вигляді підпрограм або модулів;

− програмування «знизу вгору», при цьому розробляються окремі підпрограми або модулі, з яких конструюється програма.

Вибір методу залежить від типу задачі, що розв’язується, умов розробки (один програміст або група), наявності аналогів і т.д.

Модульне програмування − це програмування, при якому будь-яка складна (велика) програма подається у вигляді сукупності логічно пов'язаних модулів.

Поняття модуля реалізовано у середовищі розробки Турбо Паскаль, починаючи з версії 4.0.

Модулі є хорошим інструментом для розробки прикладних програм, пакетів, особистих бібліотек.

6.1.2. Модуль та його структура

Модуль (UNIT-модуль, одиниця) − програмна одиниця, що автономно (окремо) компілюється та містить компоненти розділу описів (міток, констант, типів, змінних, процедур, функцій), а також може містити оператори частини, що ініцілізується.

Сам модуль не є програмою, що виконується, а призначений для використання іншими програмами і модулями.

Модуль має таку структуру:

```
UNIT <ім'я модуля>
INTERFACE
        <Розділ інтерфейсний>
IMPLEMENTATION
         <Розділ реалізації>
BEGIN
           <Розділ ініціалізації>
END.
```

Заголовок модуля складається із зарезервованого слова Unit (модуль) та імені модуля.

Ім’я модуля обирається за загальними правилами і повинно співпадати з ім’ям дискового файлу, який містить вихідний текст модуля.

Розширення в імені модуля (. pas) не вказується, воно задається за умовчанням.

Ім’я модуля використовується для його зв’язку із основною програмою за допомогою розділу Uses.

Розділ Uses може бути поміщено після заголовка модуля або за словами Interface і Implementation.

Інтерфейсна частина починається з ключового слова Interface (інтерфейс, з’єднання) і містить звернення до інших модулів та описи глобальних об’єктів, тобто міток, констант, типів, змінних і заголовків процедур і функцій, які доступні основній програмі та іншим модулям.

Розділ реалізації починається з ключового слова Implementation (виконання) і містить опис локальних для модуля об’єктів, тобто міток, констант, типів, змінних, які не доступні основній програмі та іншим модулям, і повний опис процедур та функцій. При цьому в заголовку підпрограм список формальних параметрів може бути опущений, але якщо він наводиться, то повинен точно відповідати опису в інтерфейсної частини.

Розділ ініціалізації поміщається у словесні дужки BEGIN – END і містить оператори, які будуть виконані до передачі управління

основній програмі. Це можуть бути оператори ініціалізації даних (змінних) Наприклад, оператори присвоювання, введення, а також процедури зв'язування і відкриття файлів. Розділ операторів може бути порожнім – BEGIN END або відсутнім – просто END.

В кінці модуля ставиться крапка.

Індівідуальне завдання

Похідні дані. Надані у табл. 6.1 лабораторної роботи 5.

Завдання. З використанням власних модулів скласти програму згідно з умовами лабораторної роботи 5.

Таблиця 6.1 – Варіанти індивідуальних завдань

№	I/ R	n1, n2, n3	З використанням функції визначити	З використанням процедури визначити
1	2	3	4	5
0	R	5, 4, 3	Максимальний елемент масивів	Суму та кількість від'ємних елементів масивів
1	I	4, 5, 6	Добуток додатних елементів масивів	Мінімальний елемент із від'ємних чисел, а також його порядковий номер
2	R	6, 5, 4	Суму додатних елементів масивів	Добуток та кількість елементів масивів менше ніж 1 та рівніх нулю
3	I	5, 4, 6	Суму елементів масивів неріних нулю	Максимальний та мінімальний елементи масивів
4	R	4, 3, 6	Максимальний елемент із від'ємних чисел масивів	Суму від'ємних елементів масивів та їх кількість

Продовження табл. 6.1

1	2	3	4	5
5	I	5, 6, 3	Мінімальний елемент масивів	Добуток та кількість елементів масивів нерівних нулю
6	R	6, 5, 7	Максимальний елемент із додатних чисел масивів	Максимальний елемент для елементів масивів з парними індексами, а також його порядковий номер
7	I	6, 8, 3	Суму елементів масивів з непарними індексами	Максимальний та мінімальний із від'ємних елементів масивів
8	R	8, 4, 5	Суму від'ємних елементів масивів	Мінімальний елемент масивів з непарними індексами, а також його порядковий номер
9	I	9,7,5	Максимальний елемент із парних елементів масивів	Суму та кількість додатних елементів масивів

Приклад виконання лабораторної роботи

Індивідуальне завдання. У файл **inp.txt** записати три довільних масиви дійсних чисел. Кількість елементів у масивах 9, 6, 7. З використанням функції у кожному із масивів визначити максимальний елемент серед елементів з парними індексами. З використанням процедури у кожному із масивів визначити кількість та суму додатних та від'ємних елементів. Процедури та функції реалізувати за допомогою власного модуля.

Тексти програм

```
{Текст собственного модуля lab6. Имя модуля должно
совпадать с именем файла, в котором содержится текст
программы модуля. В данном примере текст модуля должен
храниться в файле lab6.pas,после компиляции которого
создается файл lab6.tpu}
unit lab6; {Заголовок модуля lab6}
{Интерфейсный раздел модуля. Содержит описания
глобальных констант и типов данных, а также заголовки
процедур и функций, реализованных в модуле, с указанием
всех формальных параметров}
 interface
  const {Раздел описания констант}
  n1=9; {Количество элементов в первом массиве}
  n2=6; {Количество элементов во втором массиве}
  n3=7; {Количество элементов в третьем массиве}
 type {Раздел описания пользовательских типов данных}
{Количество элементов выбирается максимальным из трех}
  mas=array[1..n1] of real;
 procedure read_file(var f:text; k:integer; var m:mas);
 function max_chetn(k:integer; m:mas):real;
 procedure sum_kol(k:integer; m:mas; var sp,so:real; var
kp,ko:integer); {Окончание раздела interface}
{Раздел реализации процедур и функций модуля. В отличие
от раздела interface заголовки процедур и функций могут
указываться без формальных параметров}
 implementation
{Процедура чтения из файла inp.txt в массив. В процедуру
передаются файловая переменная f и количество элементов
в массиве k. Процедура возвращает массив m, считанный из
текстового файла.}
procedure read_file;
  var {Раздел описания локальных переменных}
   i : integer; {Параметр циклу}
begin {Начало процедуры}
{Чтение k элементов из файла и запись их в массив}
```

```
   for i:=1 to k do
    read(f,m[i]);
 end; {Конец процедуры}
{Функция определяет максимальный из элементов с четными
индексами. В функцию передаются количество элементов в
массиве k и массив m. Функция возвращает значение типа
real.}
function max_chetn;
 var {Раздел описания локальных переменных}
  max : real; {Вспомогательная переменная}
  i : integer; {Параметр циклу}
begin {Начало функции}
{Выбирается первый элемент с четным индексом}
  max:=m[2];
{Просмотр массива, начиная с третьего индекса до конца.
Проверка индекса на четность и сравнение элемента с
четным индексом с текущим значением max.}
   for i:=3 to k do
    if (i mod 2 = 0) and (m[i]>max) then
     max:=m[i];
{Присвоение имени функции вычисленного значения max.}
  max_chetn:=max;
end; {Конец функции}
{Процедура вычисляет сумму и количество положительных и
отрицательных элементов в массиве. В процедуру
передаются количество элементов в массиве k и массив m.
Процедура возвращает суммы положительных sp и
отрицательных so элементов и количества этих элементов
kp и ko соответственно}
procedure sum_kol;
 var {Раздел описания локальных переменных}
  i : integer; {Параметр циклу}
begin {Начало процедуры}
{Обнуление сумм и количеств положительных и
отрицательных элементов}
  sp:=0;
```

```
  so:=0;
  kp:=0;
  ko:=0;
   for i:=1 to k do
    begin {Начало тела циклу с параметром}
     if m[i]>0 then
      begin
{Подсчет количества и суммы положительных элементов}
       kp:=kp+1;
       sp:=sp+m[i];
      end;
    if m[i]<0 then
{Подсчет количества и суммы отрицательных элементов}
      begin
       ko:=ko+1;
       so:=so+m[i];
      end;
    end; {Конец тела циклу с параметром}
end; {Конец процедуры}
{Окончание раздела implementation}
end. {Раздел запуска модуля. Окончание модуля}

{Главная программа}
program lr6; {Заголовок программы}
{Подключение стандартного модуля crt и собственного
модуля lab6}
 uses crt, lab6;
var {Раздел описания локальных переменных}
 m1,m2,m3 : mas; {Массивы}
{Максимальные значения и суммы положительных и
отрицательных элементов для трех массивов.}
 max1,max2,max3,sp1,so1,sp2,so2,sp3,so3 : real;
{Параметр циклу i и количества положительных и
отрицательных элементов для трех массивов.}
 i,kp1,ko1,kp2,ko2,kp3,ko3 : integer;
 f : text; {f-файловая переменная}
```

```
begin{Начало программы}
 clrscr; {Очистка экрана}
{Связывание файловой переменной f с файлом inp.txt}
 assign(f,'inp.txt');
 reset(f); {Открытие файла для чтения}
{3 раза вызывается процедура read_file, реализованная в
модуле lab6, для чтения из файла и записи в массивы m1,
m2 и m3 соответственно}
 read_file(f,n1,m1);
 read_file(f,n2,m2);
 read_file(f,n3,m3);
 close(f); {Закрытие файла}
{Вывод массивов m1, m2 и m3 }
 writeln('                    massiv m1:');
  for i:=1 to n1 do
   write(m1[i]:8:2);
   writeln;
 writeln('                   massiv m2:');
  for i:=1 to n2 do
   write(m2[i]:8:2);
   writeln;
 writeln('                   massiv m3:');
  for i:=1 to n3 do
   write(m3[i]:8:2);
   writeln;
{Три раза вызывается функция max_chetn, реализованная в
модуле lab6, для определения максимальных элементов,
стоящих на четных местах,  в массивах m1, m2 и m3.
Значения, полученные с помощью функции, присваиваются
переменным max1, max2 и max3 соответственно.}
  max1:=max_chetn(n1,m1);
  max2:=max_chetn(n2,m2);
  max3:=max_chetn(n3,m3);
 writeln;
{Вывод значений переменных max1, max2 и max3}
```

```
 writeln('m1: max element s chetnym
indeksom=',max1:6:2);
 writeln('m2: max element s chetnym
indeksom=',max2:6:2);
 writeln('m3: max element s chetnym
indeksom=',max3:6:2);
 writeln;
{Три раза вызывается процедура sum_kol, реализованная в
модуле lab6, для подсчета сумм и количеств положительных
и отрицательных элементов в массивах m1, m2 и m3.
Полученные в процедуре значения возвращаются в
переменные: sp1, so1, kp1, ko1 для массива m1; sp2, so2,
kp2, ko2 для массива m2; sp3, so3, kp3, ko3 для массива
m3.}
 sum_kol(n1,m1,sp1,so1,kp1,ko1);
 sum_kol(n2,m2,sp2,so2,kp2,ko2);
 sum_kol(n3,m3,sp3,so3,kp3,ko3);
{Вывод сумм и количеств положительных и отрицательных
элементов в массивах m1, m2 и m3.}
 writeln('Massiv  Kol+   Sum+   Kol-  Sum-');
 writeln('  m1',kp1:6,sp1:10:2,ko1:4,so1:10:2);
 writeln('  m2',kp2:6,sp2:10:2,ko2:4,so2:10:2);
 writeln('  m3',kp3:6,sp3:10:2,ko3:4,so3:10:2);
 readkey; {Ожидание нажатия клавиши}
end. {Конец главной программы}
```

Схеми алгоритмів головної програми та підпрограм, вміст вхідного файлу та результати роботи програми повністю аналогічні наведеним у лабораторній роботі 5.

.

ЛАБОРАТОРНА РОБОТА 7

ЗАПИСИ В ТР

Мета роботи − набуття практичних навичок у розробці та відлагодженні програм з використанням записів.

Загальні положення

7.1. Комбіновані типи. Записи

Запис − структура даних, що складається з фіксованого числа компонентів, які називаються полями, кожен з яких може мати свій тип.

Комбіновані типи − фіксована кількість компонентів різних типів.

Записи дозволяють у компактній формі описувати різні об'єкти (списки, каталоги, картотеки, таблиці, відомості та ін.).

Розглянемо способи опису, введення/виведення і обробки записів.

7.1.1. Опис записів

При використанні записів вони повинні бути описані з використанням розділу опису або безпосередньо у розділі опису змінних.

Запис має таку структуру:

```
IZ = RECORD
        IK1 : T1;
        IK2 : T2;
        ...
        IKN : TN;
 end;
```

де

IZ – ім’я типу запису;

IK – ім’я поля (компоненти);

T − тип компоненти; будь-який (крім файлового);

RECORD - запис;

RECORD-END - операторні дужки.

Приклад опису записи:

```
Const n = 25;
Type Date = Record
               Year: integer; {Рік}
               Month: 1 .. 12; {Місяць}
               Day: 1 .. 31; {Число}
            End;
   Student = Record
               FIO: String [15]; {ПІБ}
               Dr: Date; {Дата народження}
               Str: String [20]; {Країна}
               b1,b2,b3,b4,b5: 2 .. 5; {Поле оцінок}
             End;
Vec = array [1 .. n] of Student;
Var MT_12a, MT_22a, MT_32a: Vec;
```

Порядок послідовності полів у запису не має значення.

Поле може мати кілька імен, при цьому вони розділяються комами.

Для наочності поля слід розташовувати на окремих рядках і супроводжувати коментарями.

Запис може містити варіантну частину. Наприклад, для іноземних студентів можна вказати країну.

```
Type x = (Yes, No);
   Student = Record
```

```
                    FIO: String [25]; {ПІБ}
                    Mark: 2 .. 5; {Оцінки}
                    Variant: x;
                    Case x of
                         Yes:   (Str:   String   [20]);
{Країна}
                         No: ();
               End;
```

7.1.2. Обробка записів

При обробці записів всі дії виконуються над полями запису відповідно до їх типів. Набір операцій, процедур і функцій визначається типом поля. Для запису в цілому може бути використано лише оператор присвоювання. Запис можна передавати як параметри у процедуру і функцію.

Для звернення до полів запису використовуються імена.

Складове ім'я – сукупність імені змінної типу запис та імен полів розділених крапкою.

Наприклад:

MT_12a [1]. FIO: = 'Антонов';

MT_12a [1]. Dr.Day: = 15;

MT_12a [1]. Dr. Month: = 1;

MT_12a [1]. Dr. Year: = 1986;

7.1.3. Оператор приєднання

Оператор приєднання дозволяє спростити звернення до елементів запису і має такий формат запису:

```
With <Ім'я> do
      OP;
```

де

With – з;

<Ім’я> − ім’я запису (просте чи складене);

do − робити;

ОР − оператор (простий або складовий).

7.1.4. Введення/виведення записів

При введені/виведенні записів використовуються окремі поля запису:

```
For i:=1 to n do Read (MT_12a[i].FIO,MT_12a[i].Str)
```

При використанні оператора приєднання:

```
For i:=1 to n do
    With MT_12a[i] do
        Begin
             Read (FIO);
             Read (Str);
         End;
```

Індівідуальне завдання

Похідні дані. Надані у таблиці 7.1.

Завдання. У файл **inp.txt** ввести відомість підсумків здачі сесії студентами групи. Файл повинен містити такі дані за кожним студентом: прізвище, оцінки з п’яти навчальних дисциплін, отримані в результаті складання іспитів. Скласти програму аналізу результатів здачі сесії з визначенням середнього балу кожного студента та середнього балу групи. Додаткове завдання обрати з табл. 7.1 згідно з номером варіанта.

Таблиця 7.1 – Варіанти індивідуальних завдань

Номер варіанта	Завдання
1	Визначити кількість студентів, в яких середній бал вище, ніж середній бал групи
2	Визначити студента з мінімальним середнім балом
3	Визначити номер навчальної дисципліни, в якій успішність групи є найкращою
4	Визначити кількість студентів, що мають середній бал вище, ніж 4.0
5	Визначити загальну кількість оцінок «3» для всієї групи за всіма навчальними дисциплінами
6	Визначити прізвища студенів, які мають заборгованості (оцінка «2») хоча б із однієї навчальної дисципліни
7	Визначити кількість відмінників у групі (середній бал 5.0)
8	Визначити номер навчальної дисципліни, в якій успішність групи є найгіршою
9	Визначити середній бал з третьої дисципліни
10	Визначити загальну кількість оцінок «5» для всієї групи за всіма навчальними дисциплінами
11	Відсортувати відомість у порядку зростання середнього балу
12	Визначити загальну кількість оцінок «4» для всієї групи по всіх учбових дисциплінах
13	Визначити загальну кількість оцінок «5» з іншої дисципліни
14	Визначити кількість студентів, у яких середній бал нижче, ніж середній бал групи
15	Визначити прізвища студенів, які не мають оцінок «3»
16	Визначити загальну кількість оцінок «4» з четвертої дисципліни

Приклад виконання лабораторної роботи

Індивідуальне завдання. У файл **inp.txt** ввести відомість підсумків здачі сесії студентами групи. Скласти програму аналізу результатів здачі сесії з визначенням середнього бала кожного студента, середнього бала групи і середнього бала за першою дисципліною.

Схеми алгоритмів

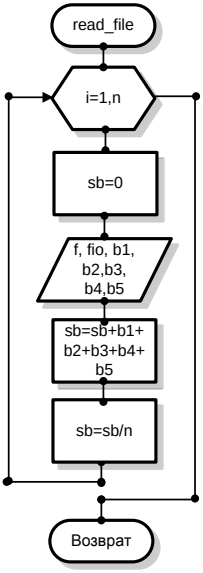


Рисунок 7.1 − Схема алгоритму процедури **read_file**

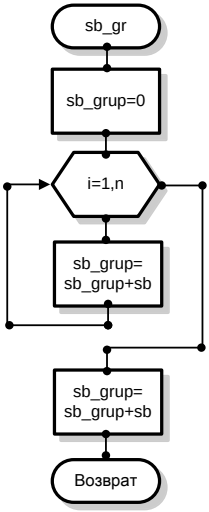


Рисунок 7.2 − Схема алгоритму процедури **sb_gr**

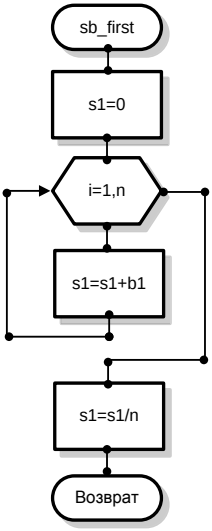


Рисунок 7.3 − Схема алгоритму процедури **sb_first**

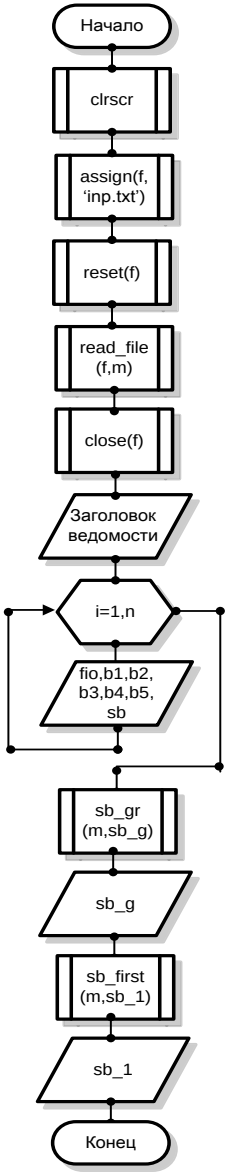


Рисунок 7.4 – Схема алгоритму основної програми

Текст програми

```
program lab7; {Заголовок программы}
 uses crt; {Подключение стандартного модуля crt}
 const {Раздел описания констант}
  n=10; {Количество студентов в группе}
 type {Раздел описания пользовательских типов
данных}
   stud=record {Запись stud}
     fio : string[10]; {Фамилия студента}
     b1,b2,b3,b4,b5 : 2..5; {Баллы по предметам}
     sb : real; {Средний балл студента}
   end;
  sp=array[1..n] of stud; {Массив записей}
{Процедура чтения данных из файла inp.txt в массив
записей с одновременным вычислением среднего балла
каждого студента. В процедуру передаются файловая
переменная f. Процедура возвращает массив записей mas.}
procedure read_file(var f:text; var mas:sp);
 var {Раздел описания локальных переменных}
  i : integer; {Параметр циклу}
begin {Начало процедуры}
{Чтение n строк из файла и запись их в массив}
  for i:=1 to n do
    begin {Начало тела циклу с параметром}
{Использование оператора with позволяет указывать
при обращении только имя поля, а имя записи указывается
в заголовке оператора with}
     with mas[i] do
      begin {Начало оператора with}
       sb:=0; {Обнуление поля среднего балла}
{Чтение из файла в текущую запись массива
mas[i]полей записи: fio, b1, b2, b3, b4, b5}
        read(f,fio);
        read(f,b1);
```

```
            read(f,b2);
            read(f,b3);
            read(f,b4);
            readln(f,b5);
            sb:=sb+b1+b2+b3+b4+b5;{Сумма оценок}
          end; {Конец оператора with}
     mas[i].sb:=mas[i].sb/5; {Вычисление среднего балла}
        end;{Конец тела циклу с параметром}
     end; {Конец процедуры}
     {Процедура вычисления среднего балла группы. В
процедуру передается массив записей mas. Процедура
возвращает средний балл группы sb_grup.}
     procedure sb_gr(mas:sp; var sb_grup : real);
      var {Раздел описания локальных переменных}
       i: integer; {Параметр циклу}
     begin {Начало процедуры}
        sb_grup:=0; {Обнуление поля среднего балла
группы}
         for i:=1 to n do
          with mas[i] do
     {Сложение текущего значения sb_grup со значением
среднего балла текущего студента sb}
           sb_grup:=sb_grup+sb;
     {Вычисление среднего балла группы}
         sb_grup:=sb_grup/n;
     end; {Конец процедуры}
     {Процедура вычисления среднего балла группы по
первому предмету. В процедуру передается массив записей
mas. Процедура возвращает средний балл группы по первому
предмету s1.}
     procedure sb_first(mas:sp; var s1 : real);
     var {Раздел описания локальных переменных}
       i: integer; {Параметр циклу}
     begin {Начало процедуры}
     {Обнуление s1}
```

```
         s1:=0;
          for i:=1 to n do
           with mas[i] do
      {Сложение баллов по первому предмету всех
студентов}
            s1:=s1+b1;
      {Вычисление среднего балла группы по первому
предмету}
          s1:=s1/n;
      end; {Конец процедуры}
      {Главная программа}
       Var {Раздел описания локальных переменных}
        m : sp; {Массив записей}
        i : integer; {Параметр циклу}
        f : text; {f-файловая переменная}
      { sb_g - средний балл группы, sb_1 - средний балл
группы по первому предмету}
        sb_g, sb_1 : real;
      begin {Начало программы}
        clrscr; {Очистка экрана}
      {Связывание файловой переменной f с файлом inp.txt}
        assign(f,'inp.txt');
        reset(f); {Открытие файла для чтения}
      {Вызов процедуры read_file для чтения из файла и
записи данных в массив m}
        read_file(f,m);
        close(f); {Закрытие файла}
      {Вывод заголовка ведомости сдачи сессии}
        writeln('Vedomost:':30);
        writeln('---------------------------------------
-');
        writeln('    Familija    Ocenki po predmetam
sr.ball');
        writeln('---------------------------------------
-');
```

```
      {Вывод исходной ведомости}
         for i:=1 to n do
          begin {Начало тела циклу с параметром}
            write(m[i].fio:15);
            write(m[i].b1:4);
            write(m[i].b2:4);
            write(m[i].b3:4);
            write(m[i].b4:4);
            write(m[i].b5:4);
            writeln(m[i].sb:6:2);
         end; {Конец тела циклу с параметром}
         writeln('----------------------------------------
-');
      {Вызов процедуры sb_gr подсчета среднего балла
группы}
         sb_gr(m,sb_g);
      {Вывод среднего балла группы}
         writeln('Srednij ball gruppi:':35,sb_g:6:2);
      {Вызов процедуры sb_first подсчета среднего балла
группы по первому предмету}
         sb_first(m,sb_1);
      {Вывод среднего балла группы по первому предмету}
         writeln('Srednij ball po pervomu
predmetu:':35,sb_1:6:2);
        readkey; {Ожидание нажатия клавиши}
      end. {Конец главной программы}
```

Вміст вхідного файлу inp.txt

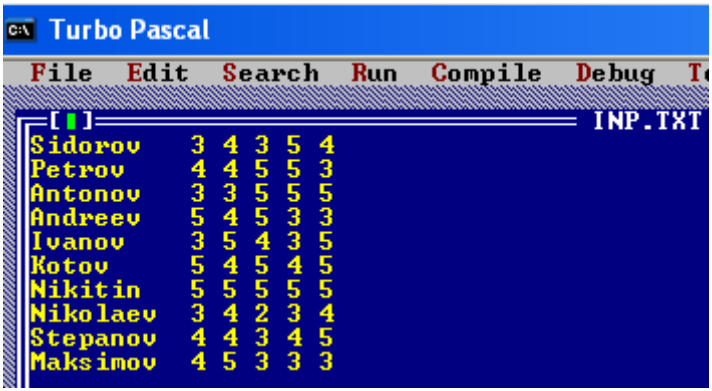


Рисунок 7.5 – Вікно із вмістом вхідного файлу **inp.txt**

Отримані результати

Рисунок 7.6 – Вікно результату запуску програми

ЛАБОРАТОРНА РОБОТА 8

СТАНДАРТНІ МОДУЛІ В ТР

Мета роботи – набуття практичних навичок у розробці та відлагодження програм з використанням стандартних модулів.

Загальні положення

8.1. Модуль GRAPH

Модуль GRAPH містить великий набір процедур і функцій, а також констант, типів і змінних, які використовуються для роботи з графічними зображеннями.

При роботі у графічному режимі елементом екрана є точка (піксель) різної світності та кольору. Координати точки змінюються від лівого верхнього кута (0, 0).

Для роботи з графікою ПК повинен містити апаратну (відеоадаптер) і програмну (графічний драйвер) підтримку.

8.1.1. Ініціалізація графічного режиму

Після включення ПК він автоматично переходить у текстовий режим, для переходу в графічний режим він повинен бути ініціалізований. Фрагмент програми ініціалізації має вигляд:

```
     Uses Crt, Graph; Var Gd, Gm, ErrCode: integer;
Begin Gd: = Detect; {Опр. типу драйвера}
     InitGraph (Gd, Gm, 'D: \ TP \ BGI'); {Шлях до
драйвера}
     ErrCode: = GraphResult;
       If ErrCode <> grOK then
          Begin
             Writeln ('Помилка ініціалізації',
GraphErroMsg (ErroeCode));
```

```
            Halt (1); {Режим не вдалося відкрити}
        End;
***
End.
```

8.1.2. Процедури і функції ініціалізації графічного режиму

Процедура ініціалізації:

InitGraph (<драйвер>, <режим>, <шлях до драйверу>);

де

– <драйвер> − змінна цілого типу (Gd);
– <режим> − режим роботи драйвера, змінна цілого типу (Gm);
– <шлях до драйверу> − змінна рядкового типу (S).

У модулі кожного типу драйвера і режиму роботи відповідають константи. Наприклад: VGA = 9; VGAHi = 2.

Якщо тип драйвера невідомий або програма розрахована на роботу з будь-яким адаптером, то можна використовувати режим автовизначення за допомогою констант Detect (код 0).

Функція GraphResult: integer − повертає код помилки при виконанні графічної операції.

Функція GraphErroMsg (<код>) − повертає текст, відповідний коду помилки.

Значення помилок зумовлені константами:

```
    Const
gr_OK = 0; {Немає помилок}
grInitGraph = - 1; {Графіка не ініцілізована}
grNotDetected = - 2; {Графічне пристрій не виявлено}
grFileNotFound = - 3; {Файл драйвера не знайдено}
grIvalidDeviceNum = - 15; {Неправильний номер
пристрою}
```

Повернення у текстовий режим

Процедура ClosetGraph − відновлює текстовий режим і вивантажує драйвер із пам'яті.

Процедура RestoreCrtMode − короткочасний вихід у текстовий режим.

Процедура SetGraphMode − встановлює новий графічний режим.

8.1.3. Координати, вікна, колір

Функції:

– GetMaxX − повертає максимальну координату X;
– GetMaxY − повертає максимальну координату Y;
– GetX − повертає поточну координату X;
– GetX − повертає поточну координату X.

Процедури:

– SetViewPort (X1, Y1, X2, Y2, <відсічення>) − встановлює графічне вікно,

де <відсічення> − логічна змінна, якщо вона дорівнює false, то межі ігноруються, в іншому випадку зображення за межами кордону відсікається;

– ClearDevice − очищення екрана;
– ClearViewPort − очищення вікна;
– SetColor − встановлює колір малювання;
– SetBkColor − встановлює колір фону;

8.1.4. Точки, лінії

Процедури:

– PutPixel (X, Y, <колір>) – виводить на екран точку з визначеними координатами та кольором;

– Line (X1, Y1, X2, Y2) – рисує лінію між заданими координатами;

– LineTo (X, Y) − рисує лінію від поточних координат до заданої точки;

– LineTRel (Dx, Dy) - рисує лінію від поточних координат до точки, заданої в приростах.

8.1.5. Дуги, кола, еліпси

Процедури:

– Circle (X, Y, R) − рисує коло;

– Arc (X, Y, <поч. кут>, <кінц. кут>, R) − рисує дугу;

– Ellipse (X, Y, <поч. кут>, <кінц. кут>, Rx, Ry) – рисує еліпс.

8.1.6. Багатокутники

Процедури:

– Rectangle (X1, Y1, X2, Y2) − рисує прямокутник;

– Bar (X1, Y1, X2, Y2) − рисує смугу;

– Bar3D (X1, Y1, X2, Y2, <глибина», <верхня грань>) − рисує паралелепіпед.

8.1.7. Завдання стилю рисування ліній

Процедура: SetLineStyle (<вигляд>, <зразок>, <товщина>)

де

<вигляд> − 0 − суцільна;
1 − точкова;
2 − штрих-пунктирна;
3 − пунктирна;
4 − визначається користувачем;

<зразок> − тільки для лінії, визначеної користувачем;

<товщина> − 1 − в один піксель;
3 − у три пікселя.

8.1.8. Зафарбовування і штрихування ділянок

Процедура: SetFillStyle (<штрихування>, <колір>).

де

<штрихування> − 0 − колір фону;
1 − суцільна;
2 − лініями;
3 − / / / / / / / / / /;
4 − \ \ \ \ товстими;
5 − / / / / товстими;
6 − \ \ \ \ \ \ \ \;
7 − + + + + + + +;
8 − x x x x x x x x;
9 − клітина;
10 − рідкісні точки;
11 − часті точки;
12 − визначається користувачем.

Процедура: FloodFill (X, Y, <колір межі>) − штрихує замкнуту область, що містить точку з координатами (X, Y) поточним кольором до заданої межі.

8.1.9. Виведення (рисування) текстів

Процедура: OutText (<текст>) − виводить текст у задане положення.

Процедура: OutTextXY (X, Y, <текст>) − виводить текст в задане положення.

Процедура: SetTextStyle (<шрифт>,<напрям>,<розмір>) − задає стиль рисування тексту;

де <шрифт> - задає тип шрифту.

При виведенні використовуються один бітовий і чотири штрихових шрифта.

Бітовий (стандартний, матричний 8 х 8 пікселів) шрифт, який входить у модуль GRAPH.TPU і містить символи як основного, так і додаткового набору ASCII.

Штрихові шрифти − символи, які формуються з векторів (штрихів) і зберігаються в окремих CHR-файлах.

Кожному стилю відповідає константа, яка описана в модулі GRAPH.TPU:

```
Const
     DefaultFont = 0; {Стандартний точковий}
     TriplexFont = 1; {Триплекс Trip.chr}
     SmallFont = 2; {Індексний Litt.chr}
     SansSriefFont = 3; {Прямий Sans.chr}
     GothicFont = 4; {Готичний Goth.chr}
```

<Напрям> − задається константою:

```
Const HorizDir = 0; {зліва направо}
      VertDir = 1; {знизу вгору}
```

<Розмір> − може змінюватися від 1 до10.

Процедура: SetTextJustify (<горизонт.>, <верт.>) − вертикальне і горизонтальне вирівнювання тексту,

де

<горизонт> = 0 − покажчик праворуч від центра;
= 1 − покажчик у центрі;
= 2 − покажчик ліворуч від центра;

<верт.> = 0 − знизу;
= 1 − зверху.

8.2. Модуль CTR

Модуль CRT містить набір процедур та функцій для управління текстовим виведенням на екран дисплея, звуковим генератором і читанням символів з клавіатури без відображення їх на екрані, а також змінних і констант режимів роботи і кольорів.

Монітор може працювати в текстовому або графічному режимі з різною роздільною здатністю.

Текстовий режим задається процедурою:

- TextMode (Co40); де TextMode − режим;
- TextMode (Co80); Bw (Black-White) − чорно-білий режим;
- TextMode (Bw40); 40 і 80 − символів в рядку;
- TextMode (Bw80).

Параметри процедури можна задати за допомогою констант визначених у модулі

```
Const
    Bw40 = 0;
    Co40 = 1;
    Bw80 = 2;
    Co80 = 3;
```

Один із режимів установлюється за умовчанням при настройці системи.

Для завдання кольору символів і фону використовуються такі константи:

```
   Const
{Для кольору символів і фону}
         Black = 0; {Чорний}
         Blue = 1; {Синій}
         Green = 2; {Зелений}
         Cyan = 3; {Блакитний}
         Red = 4; {Червоний}
         Magenta = 5; {Фіолетовий}
```

```
          Brown = 6; {Коричневий}
          LightGray = 7; {Світло-сірий}
{Для кольору символів}
          DarkGray = 8; {Темно-сірий}
          LightBlue = 9; {Яскраво-синій}
          LightGreen = 10; {Яскраво-зелений}
          LightCyan = 11; {Яскраво-блакитний}
          LightRed = 12; {Рожевий}
          LightMagenta = 13; {Малиновий}
          Yellow = 14; {Жовтий}
          White = 15; {Білий}
          Blinck = 128; {Мерехтіння символу}
```

Працювати можна на всьому екрані 80 х 25, або виділити вікно за допомогою процедури Window (X1, Y1, X2, Y2). За умовчанням встановлюється вікно (1, 80, 1, 25).

Для явного виділення вікна після процедури Window необхідно виконати TextBackGround і ClrScr.

Таблиця 8.1 – Основні процедури модуля CRT

№	Процедура	Призначення
1	2	3
1	AssignCRT	Текстове виведення на екран
2	ClrScr	Очищення екрана, курсор у першу позицію
3	ClrEol	Видаляє символ від курсора до кінця рядка
4	DelLine	Видаляє рядок, в якому знаходиться курсор
5	InsLine	Вставляє рядок у позицію, в якій знаходиться курсор
6	GoToXY (X, Y)	Переміщає курсор у задану позицію

Продовження табл.8.1

7	Delay (n)	Припиняє виконання програми на n мілісекунд
8	NormVideo	Установлює нормальну яскравість символів
9	LowVideo	Установлює знижену яскравість символів
10	HignVideo	Установлює підвищену яскравість символів
11	Sound (f)	Включає звуковий генератор (f-Гц)
12	NoSound	Вимикає звуковий генератор
13	TextMode	Установлює текстовий режим
14	TextColor (c)	Задає колір символів
15	TextBackGround	Задає колір фону
16	Window (X1, Y1, X2, Y2)	Створює текстове вікно

Таблиця 8.2 – Основні функції модуля CRT

№	Процедура	Призначення
1	WhereX	Повертає горизонтальну координату (стовпець)
2	WhereY	Повертає вертикальну координату (рядок)
3	KeyPressed	Повертає true, якщо натиснута клавіша і false в іншому випадку.
4	ReadKey	Читає символ із клавіатури без відображення на екрані

Індивідуальне завдання

Для модуля CRT

Вивести на екран текстове повідомлення. Зміст, колір та розташування тексту на екрані обрати із табл. 8.3 згідно з номером варіанта.

Таблиця 8.3 – Варіанти індивідуальних завднь

№	Зміст текстового повідомлення	Колір	Розташування на екрані
1	Прізвище	Blue	У центрі екрана
2	Ім’я	Green	У лівому нижньому куті екрана
9	Група	Cyan	У правому верхньому куті екрану
4	Навчальна дисципліна	Red	У лівому верхньому куті екрана
5	Факультет	Magenta	У правому нижньому куті екрана
6	Прізвище	Brown	У середині лівої верхньої чверті екрана
7	Ім’я	Lightgray	У середині правої нижньої чверті екрана
8	Група	Lightblue	У середині лівої нижньої чверті екрана
9	Навчальна дисципліна	Lightgreen	У центрі екрану
10	Факультет	Lightcyan	У лівому нижньому куті екрана
11	Прізвище	Lightred	У правому верхньому куті екрана
12	Ім’я	Lightmagenta	У лівому верхньому куті екрана
13	Група	Yellow	У правому нижньому куті екрана
14	Навчальна дисципліна	White	У середині лівої верхньої чверті екрана
15	Факультет	Blue	У середині правої нижньої чверті екрана
16	Прізвище	Green	У середині лівої нижньої чверті екрана

Для модуля GRAPH

Вивести на екран графічний примітив. Тип примітива, колір та розташування його на екрані обрати із таблиці згідно з номером варіанта.

Таблиця 8.4 − Варіанти індивідуальних завдань

№	Графічний примітив	Колір	Розташування на екрані
1	Circle	Blue	У центрі екрана
2	Arc	Green	У лівому нижньому куті екрана
3	Bar	Cyan	У правому верхньому куті екрану
4	Bar3d	Red	У лівому верхньому куті екрана
5	Rectangle	Magenta	У правому нижньому куті екрана
6	Text	Brown	У середині лівої верхньої чверті екрана
7	Circle	Lightgray	У середині правої нижньої чверті екрана
8	Arc	Lightblue	У середині лівої нижньої чверті екрана
9	Bar	Lightgreen	У центрі екрану
10	Bar3d	Lightcyan	У лівому нижньому куті екрана
11	Rectangle	Lightred	У правому верхньому куті екрана
12	Text	Lightmagenta	У лівому верхньому куті екрана
13	Circle	Yellow	У правому нижньому куті екрана
14	Arc	White	У середині лівої верхньої чверті екрана
15	Bar	Blue	У середині правої нижньої чверті екрана
16	Bar3d	Green	У середині лівої нижньої чверті екрана

Приклад виконання лабораторної роботи

Індивідуальне завдання. Здійснити горизонтальний рух зліва направо та скролінг текстового повідомлення із випадковим кольором тексту. По вертикалі текст розмістити всередині екрана.

Схема алгоритму:

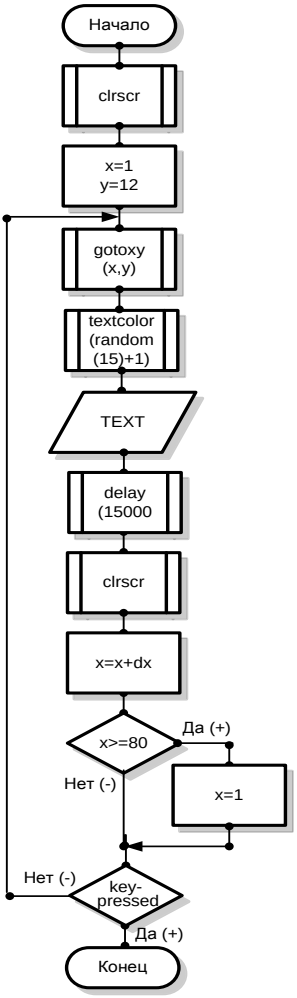


Рисунок 8.1 – Схема алгоритму програми

Текст програми

```
program lab8_1; {Заголовок программы}
uses crt; {Подключение стандартного модуля crt}
const {Раздел описания констант}
  dx=1; {Шаг движения текста по горизонтали}
  var {Раздел описания переменных}
{Координаты курсора по горизонтали и вертикали}
   x,y : integer;
begin {Начало программы}
  clrscr; {Очистка экрана}
{Начальное положение текста на экране: крайняя
левая позиция по горизонтали (x = 1) и середина экрана
по вертикали (y = 12)}
  x:=1;
  y:=12;
  repeat {Начало тела циклу с постусловием. Цикл
выполняется до нажатия любой клавиши}
{Перевод курсора в позицию с координатами x,y}
   gotoxy(x,y);
{Выбор цвета текста случайным образом}
   textcolor(random(15)+1);
   writeln('TEST');{Вывод текста}
   delay(15000);{Задержка}
   clrscr;{Очистка экрана}
   x:=x+dx;{Сдвиг текста на одну позицию вправо}
{Если текущая позиция курсора по горизонтали
выходит справа за пределы экрана, то установить курсор в
крайнюю левую позоцию}
   if x>=80 then x:=1;
  until keypressed; {Конец тела циклу с
постусловием}
 readkey; {Ожидание нажатия клавиши}
```

```
end. {Конец программы}
```

Отримані результати

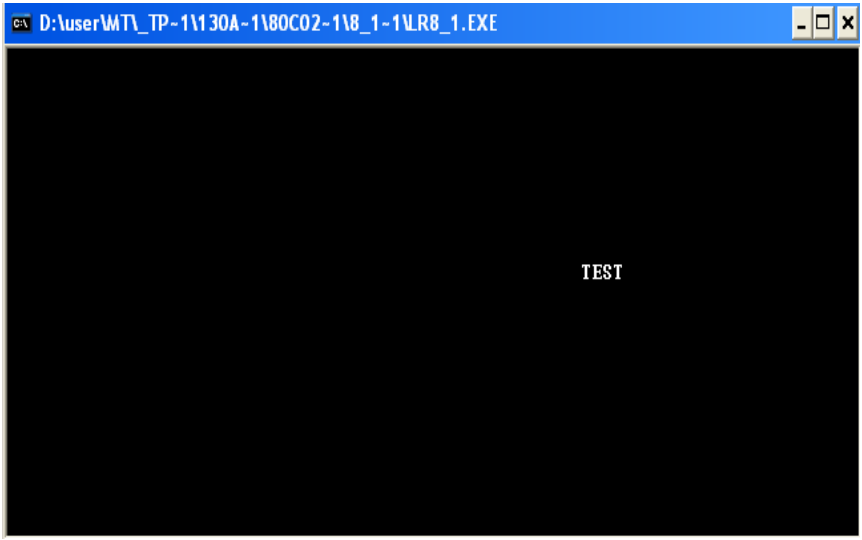


Рисунок 8.2 – Вікно результату запуска програми

Індивідуальне завдання. У графічному режимі вивести текстове повідомлення в центрі екрана. Здійснити рисування випадковим кольором кола із випадковими координатами центра та випадковим радіусом.

Схема алгоритму:

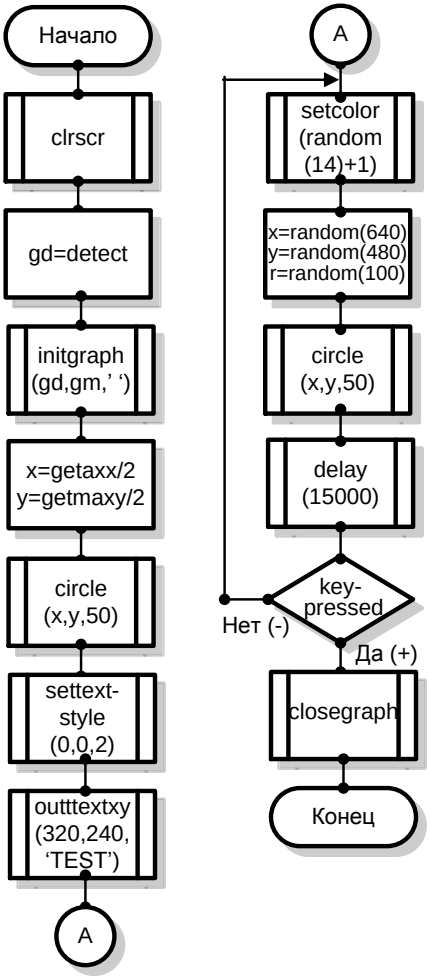


Рисунок 8.3 – Схема алгоритму програми

Текст програми

```
program lab8_2; {Заголовок программы}
{Подключение стандартных модулей crt и graph}
 uses crt, graph;
 var {Раздел описания переменных}
{gd-тип монитора, gm-разрешающая способность
экрана, x,y-координаты центра окружности, r-радиус
окружности}
  gd,gm,x,y,r : integer;
begin {Начало программы}
 clrscr; {Очистка экрана}
  gd:=detect;{Автоопределение типа монитора}
{Инициализация графического режима}
  initgraph(gd,gm,'');
{Выбор координат точки в центре экрана}
  x:=getmaxx div 2;
  y:=getmaxy div 2;
{Рисование окружности в центре экрана}
  circle(x,y,50);
{Установка стиля выводимого текста}
  settextstyle(0,0,2);
{Вывод текста в графическом режиме в центре экрана}
  outtextxy(getmaxx div 2, getmaxy div 2, 'TEST');
  repeat {Начало тела циклу с постусловием. Цикл
выполняется до нажатия любой клавиши}
{Выбор цвета рисунка случайным образом}
   setcolor(random(14)+1);
{Выбор координат центра и радиуса окружности
случайным образом}
   x:=random(640);
   y:=random(480);
   r:=random(100);
{Рисование окружности}
   circle(x,y,r);
```

```
         delay(10000); {Задержка}
        until keypressed; {Конец тела циклу с
постусловием}
        readkey; {Ожидание нажатия клавиши}
      {Закрытие графического режима и возврат в текстовый
режим}
        closegraph;
      end.{Конец программы}
```

Отримані результати

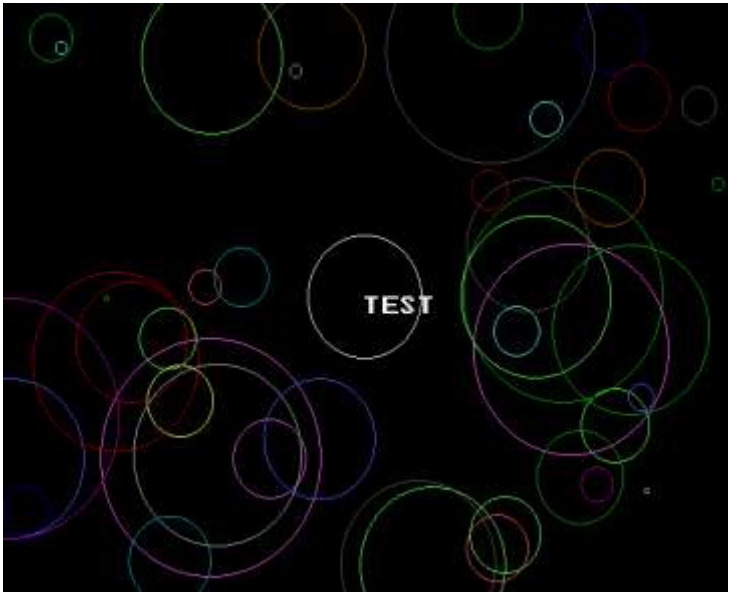


Рисунок 8.4 – Вікно результату запуску програми

Список літератури

1. В.В. Фаронов. Турбо Паскаль 7.0 Начальный курс: учеб. пособ. – М.: Издательство “ОМД Групп”, 2003. − 616 с.

2. І.О. Фурман, В.А.Краснобаєв, В.Д.Далека, Н.А. Корольова, О.М. Рисований. Моделі та структура даних у системах автоматизованого керування: Підруч. для ВНЗ/М-во освіти і науки України. – К., 2004. – 253 с.

3. И. Г. Семакин, А. П. Шестаков. Основы программирования**:** учеб. для студ. образов. учрежд. среднего профессионального образования. – М.: Академия, 2008. − 124 с.

4. И. Г. Хусаинов, Г. Я. Хусаинова, С. Л. Хасанова. Язык программирования Турбо Паскаль: учеб. пособ. − Уфа: РИЦ БашГУ, 2008. – 248 с.

5. Д.Б. Поляков, И.Ю. Кругляков. Программирование в среде Турбо-Паскаль. − М.: МАИ, 1992. – 576 с.

6. О.П. Зеленяк. Практикум программирования на Turbo Pascal. Задачи, алгоритмы, решения. Серия: Самоучитель. − И.: ДМК пресс, 2007. − 312 с.

Зміст

Навчальне видання

ЧЕРНИХ Олена Петрівна
ШЕЇН Олександр Миколайович

ІНФОРМАТИКА.
ЧАСТИНА 1

Лабораторний практикум

для студентів
6.050502 «Інженерна механіка»,
6.050403 «Інженерне матеріалознавство»

Роботу до друку рекомендував *В.Д. Дмитрієнко*
Редактор *Н.В.Верстюк*

План 2015 р., поз. 94

Підп. до друку _____ Формат 60 х 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 7,5. Наклад 100 прим.
Зам № 216. Ціна договірна.

Видавничий центр НТУ "ХПІ".
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ "ХПІ", 61002, Харків, вул. Фрунзе, 21